THESE

présentée à

L' UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

pour l'obtention du

DOCTORAT TROISIEME CYCLE

Spécialité : MATHEMATIQUES

Mention : INFORMATIQUE

par MONSIEUR ABDELKADER LAHYANE

Sujet de la thèse : MISE EN OEUVRE D'UNE APPLICATION DE TRAITEMENT DE SPECTRE DE FOURIER À BASE DE MICROPROCESSEURS

Soutenue le 4 NOVEMBRE 1982 devant la Commission

composée de : Président MME LE PROFESSEUR F. MADAULE

Examinateurs : MM. LE PROFESSEUR M. BAUDRY
L. HENRY
P. ZANETTACCI
A. VALENTIN

THESE

présentée à

L' UNIVERSITE PIERRE ET MARIE CURIE (PARIS VI)

pour l'obtention du

DOCTORAT TROISIEME CYCLE

Spécialité: MATHEMATIQUES

Mention: INFORMATIQUE

par MONSIEUR ABDELKADER LAHYANE

Sujet de la thèse: MISE EN OEUVRE D'UNE APPLICATION DE TRAITEMENT DE SPECTRE DE FOURIER À BASE DE MICROPROCESSEURS

Soutenue le 4 NOVEMBRE 1982 devant la Commission

composée de: Président MME LE PROFESSEUR F. MADAULE

Examinateurs: MM. LE PROFESSEUR M. BAUDRY
L. HENRY
P. ZANETTACCI
A. VALENTIN

A ma mère et mon épouse.

J'exprime à Madame F. MADAULE, Professeur à l'Université Pierre et Marie CURIE (PARIS VI), toute ma gratitude pour avoir accepté la présidence du Jury.

Ma profonde reconnaissance va à Monsieur P. ZANETTACCI, Assistant à l'Université Pierre et Marie CURIE, pour m'avoir orienté dans la recherche et pour s'être intéressé à ce travail.

Tous mes remerciements vont à Monsieur M. BAUDRY, Professeur à l'Université du Maine (Le Mans) pour avoir bien voulu faire partie du Jury.

Que Monsieur L. HENRY, Directeur de Recherches au C.N.R.S., trouve ici l'expression de ma respectueuse gratitude pour avoir bien voulu m'accueillir dans son laboratoire et pour le temps qu'il a bien voulu me consacrer.

Je remercie très vivement Monsieur A. VALENTIN, Chargé de Recherche au C.N.R.S. pour m'avoir aidé tout au long de ce travail avec une sympathie constante.

Je remercie très chaleureusement Mesdames N. DAGENS et O. FELLMANN qui ont assuré la frappe de cette thèse avec une attention et une gentillesse particulière.

Que tous les membres du Laboratoire de Spectronomie Moléculaire et du Laboratoire Système logiciels et matériels, techniciens et chercheurs soient enfin remerciés pour leur sympathie à mon égard.

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41543523_0035

TABLE DES MATIERES

	Pages
INTRODUCTION	1
CHAPITRE I	
Introduction	10
I - PRINCIPES DE LA SPECTROMETRIE PAR TRANSFORMATION DE FOURIER.	12
II - TRANSFORMATION DE FOURIER	21
III - ALGORITHME DE TRANSFORMATION DE FOURIER RAPIDE (TFR)	25
IV - ADAPTATION DE L'ALGORITHME DE COOLEY-TUCKEY A NOTRE CAS.	29
V - CONVOLUTION	32
VI - UTILISATION DE LA TFR DANS LE CALCUL DE LA CONVOLUTION ET DE LA CORRELATION.	33
Conclusion	36
CHAPITRE II	
Introduction	38
I - CIRCUIT INTEGRE SPECIALISE DANS LES TRANSFORMATIONS DE FOURIER RAPIDES	42
II - PARALLELISATION DE LA TRANSFORMATION DE FOURIER RAPIDE	48
III - ALGORITHME PARALLELE POUR UN GRAND NOMBRE DE MESURES.	58
Conclusion	61
CHAPITRE III	
Introduction	63
I - CALCUL DIRECT (BASSE RESOLUTION)	64
II - CALCUL D'UNE TRANCHE SPECTRALE (Haute résolution)	70
III - NOTIONS SUR LA RECHERCHE DE RAIES DANS LE SPECTRE	73
IV - CONSIDERATIONS SUR LA RECHERCHE PARALLELE DES RAIES	75

V - CONSIDERATIONS GENERALES SUR LE BRUIT DANS LA METHODE DE TRAITEMENT DES SPECTRES PAR LA TFR 76

VI - EVALUATION DES PERFORMANCES GLOBALES DU SYSTEME D'ACQUISITION DE MESURES. 78

Conclusion 82

CHAPITRE IV

Introduction 83

I - ELEMENTS DU SYSTEME 84

II - LE SOUS-SYSTEME SS1 90

III - LE SOUS-SYSTEME D'ENTREE-SORTIE SS2 104

IV - MISE EN OEUVRE DES PROGRAMMES DE L'APPLICATION 111

Conclusion 114

CONCLUSION 115

ANNEXE

1ère PARTIE 123

2ème PARTIE 133

3ème PARTIE 170

REFERENCES 181

INTRODUCTION

L'application de l'informatique à des domaines de recherche fondamentale tel que la spectronomie moléculaire s'est traduite par l'utilisation des gros ordinateurs centralisés. Cette centralisation a imposée depuis très longtemps des sauvegardes de données sur des rubans perforés et des bandes magnétiques. Le transport des rubans et des bandes magnétiques, l'éloignement et la non disponibilité des moyens de calcul ont imposé des contraintes d'expérimentation rigoureuses telles que le nombre d'expériences limité dans la journée, double bande d'enregistrement pour une même expérience.

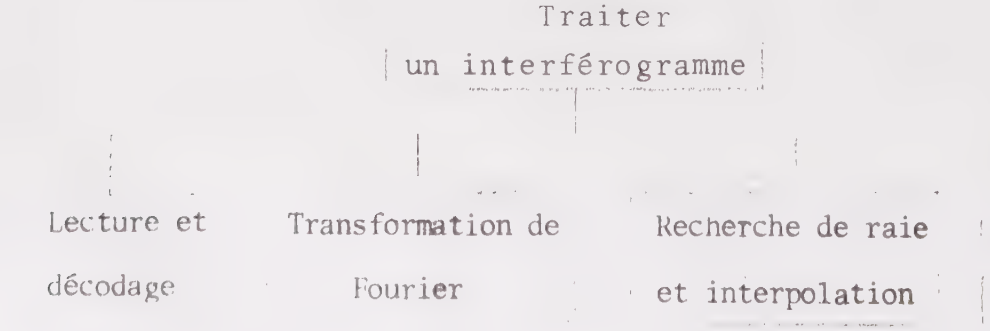
Depuis la construction d'un interféromètre de Michelson en 1959 au Laboratoire Aimé Cotton jusqu'en 1969, le logiciel et le matériel évoluaient en fonction de l'environnement et les moyens de calcul. Le nombre de points de l'enregistrement augmentaient très rapidement alors que les versions des calculateurs variaient lentement. Les temps de calcul diminuaient alors que les temps d'enregistrement augmentaient. Le tableau I montre l'évolution des caractéristiques du matériel et logiciel mis en oeuvre de 1959 à 1973

Tableau I

Année	1959-60	1964-66	1965-68	1969-73	1973-79	
Différence de marche (cm)	1	10	160	>160	>1000	
Limite de résolution (cm^{-1})	1	0,1	6.10^{-3}	$<6.10^{-3}$	$<10^{-3}$	
Nombre d'échantillons $x10^3$	10	64	64	$\simeq 2.10^3$	$>2.10^3$	
Nombre calculé par la transformation de Fourier $x\ 10^3$	1,5	64	64	$\simeq 2.10^3$	$>2.10^3$	16
Modèle de calculateur	IBM7040	IBM7040	IBM7040	IBM370	AMDHAL	IRI 80
Temps d'exécution (mn)	300	105	105	8*	<1	≃0,

* Le même calcul avec une transformation de Fourier classique durerait 4 ans.

Les temps de calcul augmentaient avec l'introduction d'autres programmes pour le traitement proprement dit du spectre de Fourier. Le chemin du calcul suivi par l'utilisateur était toujours le même. Il consistait à enregistrer les données sur une bande magnétique, l'envoyer au centre de calcul pour le traitement. Ce traitement est constitué par une lecture et le décodage de la bande, un calcul de transformation de Fourier classique ou rapide, l'interpolation et la recherche des raies dans le spectre interpolé. Ce schéma se traduit par la figure suivante :



Différentes composantes de ces modules existent et peuvent être utilisés en fonction des besoins de l'utilisateur. Pour faciliter l'utilisation de ces programmes par les chercheurs non informaticiens, il a été nécessaire d'automatiser le chargement et l'exécution par une réduction de cartes contrôles et des "overlay". Ainsi, en 1973, l'amélioration des logiciels était à sa limite et les besoins de l'utilisateur étaient satisfaits. L'objectif de calcul et de traitement du spectre était atteint alors que l'évolution des interféromètres de Michelson est loin de sa limite de perfectionnement. Cependant les logiciels ont été étudiés pour atteindre quatre objectifs :

- un premier objectif a été, d'une part de rendre le temps de calcul de la transformation de Fourier pratiquement négligeable vis à vis de la durée totale des calculs nécessaires à l'exploitation proprement dite du spectre, et d'autre part de permettre ce calcul indépendamment de la taille de la mémoire centrale de l'ordinateur dont on dispose.

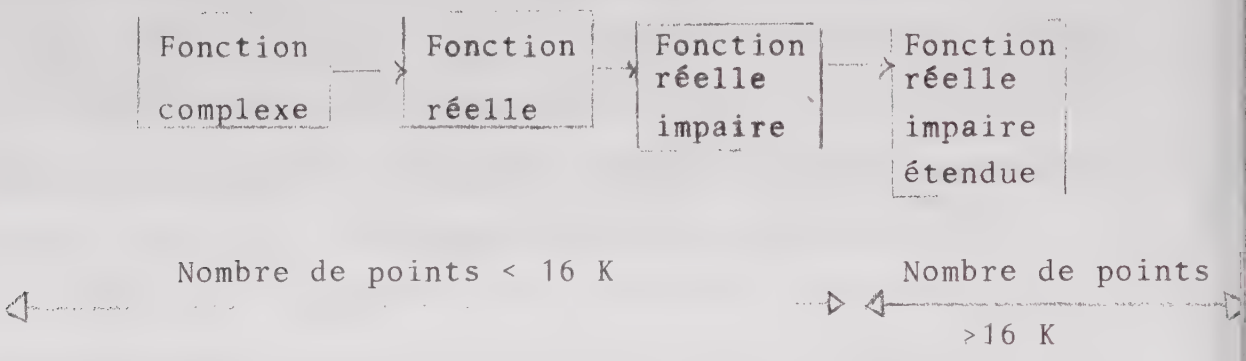
- un deuxième objectif a été de réduire la durée de l'interpolation du spectre calculé par la transformation de Fourier.

- un troisième objectif a été d'améliorer toute la partie acquisition de données sur trois plans : la fiabilité, la capacité de stockage et la rapidité tant au stade de l'enregistrement que de la lecture et du décodage des valeurs de l'enregistrement.

- un quatrième objectif plus général a été d'améliorer l'exploitation de ces programmes par les non informaticiens.

La réduction du temps de calcul de transformation de Fourier est, jusqu'à 1982, faite en deux temps :

Premier temps : l'apparition de l'algorithme de Cooley-Tuckey en 1965 [12] a contribué largement à la diminution du temps de calcul. D'autres algorithmes, dérivés de l'algorithme de Cooley-Tuckey sont apparus après 1965 [24],[11],[6],[5],[8]. Certains algorithmes [14] sont apparus en 1973 et adaptés à des fonctions réelles impaires et des fonctions réelles paires. Le chemin suivi était, jusqu'à présent,le suivant :



Deuxième temps : la transformation de Fourier rapide est utilisée dans de vastes domaines de traitement digital des signaux. Les algorithmes utilisés pour mettre en oeuvre cette transformation exigent des calculs arithmétiques intensifs aussi bien que des contrôles complexes. Le problème était de trouver une architecture de machine capable de supporter l'algorithme de la transformation de Fourier rapide. La solution était en partie trouvée [13], en introduisant un processeur spécialisé en un seul circuit intégré. Mais ce circuit est limité à 32 points. On peut déduire une nouvelle architecture pour 1024 points en utilisant le circuit 32 points. Ce circuit suppose l'existence de valeurs des sinus et cosinus stockées dans des ROM et des RAM pour stocker les résultats intermédiaires. Un autre circuit intégré [17] est mis au point comme séquenceur d'adresse pour la transformation de Fourier rapide. Ce séquenceur, asscié avec une unité arithmétique et logique rapide peut donner une bonne performance pour

le calcul de la transformation de Fourier rapide.

La rigidité d'utilisation de ces circuits et leur modèle figé ne permet pas d'aller au delà de 1024 points. Les temps de calcul dépassent largement les systèmes utilisés jusqu'en 1979.

Ces circuits permettent la diminution du temps de calcul de l'algorithme de Cooley-Tukey sans autant apporter des améliorations sur le reste de l'algorithme adapté à une fonction réelle impaire.

D'autres solutions peuvent apparaître satisfaisantes dans une partie de l'algorithme telles que les opérateurs arithmétique et logique rapide et les fonctions sinus et cosinus cablées.

Mon travail est essentiellement la mise en oeuvre de méthode nouvelle de traitement de l'interférogramme. Un objectif qui n'a pas été étudié dans les travaux d'avant 1973, est la prédiction du bon fonctionnement de l'expérience. En effet, l'éloignement et la durée de l'enregistrement supposent les conditions de l'expérience et l'enregistrement cohérents. Ceci n'étant valable que dans le cas où on est assuré des bons réglages. Perdre une journée dans l'enregistrement était un handicap sérieux.

J'ai donc à revoir les quatre objectifs pour donner une solution satisfaisante au cinquième objectif. Ainsi, je suis reparti au point de départ pour reprendre l'ensemble des étapes de l'acquisition des interférogrammes au traitement du spectre.

J'ai décomposé le problème en deux parties distinctes:

1- comprendre le langage des physiciens, utilisations.

2- Definir le produit et l'adapter à leur problème. Le but

est donc essentiellement d'avoir une application adaptée aux besoins de l'utilisateur. Mon travail est constitué de trois parties :

1- Electronique

2- Algorithmes et programmation

3- Mise en oeuvre

L'acquisition des données de l'interféromètre de Michelson a nécessité un interface. Cet interface est composé de :

1- un coupleur électronique

2- une carte microprocesseur

3- un logiciel de base

La contrainte principale d'acquisition est la vitesse de transmission des mesures.

Dans le cas où l'interféromètre débite à des vitesses supérieures à 100 mesures à la seconde, le système l'interrompt pour sauvegarder les mesures sans en perdre, ensuite il continue l'acquisition. Cette gestion de l'interféromètre est réalisée par la carte microprocesseur et le logiciel de base.

Le traitement de l'interférogramme comporte :

- calcul du spectre
- son interpolation
- sa visualisation.

Cette thèse comprend quatre chapitre et une annexe. Dans le premier chapitre, nous décrivons l'interféromètre de Michelson et la méthode mathématique qui donne un aperçu sur la fonction à calculer. Par des moyens divers que nous n'exposons pas dans ce chapitre, en raison de leur longueur, on obtient une fonction réelle impaire que l'on appelle interférogramme. Cette fonction est une transformation de Fourier, sa transformation de Fourier inverse nous donne alors le spectre cherché. L'algorithme de Cooley-Tuckey [12] est réadapté à notre cas pour calculer la transformation de Fourier inverse de l'interférogramme impair [14].

Dans le deuxième chapitre, nous nous efforçons de faire une critique théorique des circuits intégrés à haute intégration [13] [17] qui calcule la transformation de Fourier rapide par l'algorithme de Cooley-Tuckey. On décrit un algorithme théorique adapté à un nombre très élevé de points de l'interférogramme [14]. On n'aborde pas l'aspect très théorique du traitement parallèle de l'algorithme qui supposerait un temps deux à trois fois plus que le temps mis dans l'application. Néanmoins le découpage qu'on propose pour résoudre le problème de la taille mémoire peut apporter une aide non négligeable au calcul parallèle de la transformation de Fourier rapide adaptée à l'interférogramme impair.

Dans le troisième chapitre, on décrit la méthode d'interpolation [9] et d'apodisation du spectre. En effet, le spectre résultant de la transformation de Fourier est illisible, les pieds des raies ne sont pas atténués, enfin des parties du

spectre de Fourier avant l'interpolation renferme d'autres informations à cause du calcul à basse résolution. L'interpolation par bloc de 1024 points, pour avoir comme résultat des blocs de 5120 points, est historiquement admise comme étant la plus adaptée aux applications proches de la nôtre [14] . Le choix des fonctions d'apodisation décrites dans ce chapitre permettent de mieux atténuer les effets de bord des raies du spectre interpolé.

Dans le quatrième chapitre, on explique l'application que nous avons réalisée au laboratoire de Spectronomie Moléculai conjointement avec l'Institut de Programmation , notamment l'équipe L.S.I. du Laboratoire Systèmes Logiciels et Matériels. Ce chapitre constitue une première approche de la solution du problème posé par cette application.

L'application développée dans ce chapitre est composée de deux parties :

- la première partie concerne le couplage par un micro-ordinateur de l'interféromètre de Michelson.
- la deuxième partie décrit la programmation des algorithmes de la gestion de l'interféromètre, de transmission des données, de la mise en forme des mesures, et du traitement de l'interférogramme impair (adaptation de l'algorithme de Cooley-Tuckey, interpolation et apodisation).

On a donc réalisé un système autour de deux micro-ordinateurs 8 bits et d'un opérateur mathématique pouvant évolue sur site ; ceci résout le problème de l'augmentation de puissanc sans remise en cause des investissements. Cette fonctionnalité essentielle pour le laboratoire de spectronomie moléculaire étai prévue dès le début de la conception de cette application.

La partie annexe est composée d'algorithmes des programmes mis au point dans l'application. Cette annexe a trois parties :

1ère partie : Algorithmes des programmes de la transformation de Fourier rapide définie par Cooley-Tuckey

2ème partie : Imprimés des programmes de l'application qui fonctionnent actuellement dans le système.

3ème partie : Symboles utilisés et leur signification ainsi que les commandes utilisées pour la mise en route de l'application.

CHAPITRE I

INTRODUCTION

La spectroscopie par transformation de Fourier, qui permet de s'approcher des conditions idéales d'analyse du rayonnement émis par une source, à savoir utiliser pendant toute la durée de l'enregistrement la totalité des informations pour tous les éléments spectraux, s'est progressivement imposée au cours des deux dernières décennies. A la suite des travaux de Jacquinot et Felgett sur les facteurs de mérite des différents spectromètres, le spectromètre de Fourier construit au laboratoire de spectronomie moléculaire répond aux besoins d'analyse dans l'infrarouge moyen (3 µm à 20 µm puis 100 µm) des spectres moléculaires.

Ce domaine spectral contient les premiers niveaux de vibration dont la connaissance précise est indispensable à la physique moléculaire.

Dans ce chapitre, après une description sommaire du spectromètre, nous rappellerons comment le signal enregistré interférogramme) après analyse harmonique permet de reconstituer le spectre enregistré. Enfin, nous décrirons les méthodes utilisées en spectrométrie pour calculer la transformation de Fourier discrète.

L'interférogramme étant impair, l'algorithme de Cooley-Tuckey a été réadapté pour calculer la transformée de Fourier. Cet algorithme que j'ai introduit dans notre application

a pour but de calculer et d'adapter l'algorithme de transformation de Fourier rapide à notre cas précis dans lequel on enregistre une fonction réelle impaire.

I - PRINCIPES DE LA SPECTROMETRIE PAR TRANSFORMATION DE FOURIER

I-1 Interféromètre de Michelson

L'interféromètre de Michelson forme à partir d'une source S deux sources secondaires S_1 et S_2. Le déplacement du miroir mobile fait varier la différence de marche entre les deux sources. On peut donc considérer que l'on dispose de deux sources, émettant le même signal, situées à la distance Δ, l'une par rapport à l'autre.

Pour permettre à la différence de marche de varier sans que l'appareil se dérègle on remplacera les miroirs classiques de l'interféromètre de Michelson par des systèmes réfléchissants tels qu'une rotation ne modifie pas le trajet optique au premier ordre. Le dispositif utilisé est constitué par un miroir concave et d'un miroir plan placé au foyer du miroir concave (oeil de chat).

Considérons une source S_o, monochromatique de longueur d'onde λ_o et de flux incident Φ_o. Le flux émergent Φ_e, en négligeant les pertes de lumière, s'écrit en fonction de Φ_o et λ_o:

$$\Phi_e = \Phi_o \cos^2 \frac{\pi\Delta}{\lambda_o} \qquad \text{pour une source ponctuelle.} \quad (1)$$

Si on note $\sigma_i = \lambda_o^{-1}$, le nombre d'ondes du faisceau, alors

$$\Phi_e = \frac{1}{2} \Phi_o(1 + \cos 2\pi \sigma_i \Delta). \quad (2)$$

Supposons maintenant l'interféromètre éclairé par un faisceau dont on notera $\Phi_\sigma(\sigma)$ le flux monochromatique

incident correspondant au nombre d'ondes σ.

Alors le flux émergent s'écrit :

$$\Phi(\Delta) = \int_{-\infty}^{+\infty} \Phi_\sigma(\sigma)(1 + \cos 2\pi\sigma\Delta)d\sigma \qquad (3)$$

pour une source étendue (faisceau lumineux).

$\Phi_\sigma(\sigma)$ étant symétrique par rapport à l'origine on peut écrire :

$$\Phi(\Delta) = \frac{1}{2}\int_{0}^{+\infty} \Phi_\sigma(\sigma)(1 + \cos 2\pi\sigma\Delta)\, d\sigma \qquad (4)$$

dont la partie dépendant de Δ : $\Phi_1(\Delta) = \frac{1}{2}\int_0^{+\infty} \Phi_\sigma(\sigma)\cos 2\pi\sigma\Delta d\sigma$

La fonction $F(\Delta) = \Phi_1(\Delta)$ quelle que soit la valeur de Δ comprise entre 0 et ∞ est appelée l'interférogramme. Il suffit donc de calculer la transformée de Fourier de l'interférogramme pour obtenir la fonction $\Phi_\sigma(\sigma)$.

On obtient donc :

$$\Phi_\sigma(\sigma) = \int_{0}^{+\infty} F(\Delta) \cos 2\pi\sigma\Delta d\Delta \qquad (5)$$

On mesure en réalité une intensité lumineuse $\mathscr{I}(\Delta)$ définie au coefficient près par :

$$\mathscr{I}(\Delta) = \int_{0}^{+\infty} B_\sigma(\sigma) \cos 2\pi\sigma\Delta d\sigma \qquad (6)$$

où $B_\sigma(\sigma)$ est le spectre étudié.

Ainsi pour une différence de marche donnée, on recueille sur le détecteur un signal contenant toutes les informations du spectre ("multiplexage"). Ainsi l'énergie contenue dans chaque élément spectral est intégrée pendant le temps T de la mesure alors qu'avec les méthodes séquentielles, l'énergie

n'est intégrée que pendant le temps T/N si N est le nombre d'éléments spectraux analysés.

Le gain sur le rapport signal sur bruit dépendra de la nature du détecteur, si celui-ci possède un bruit propre indépendant du signal (cas du détecteur infrarouge) alors le gain est égal à $\sqrt{N}$ (alors que dans le cas d'un photomultiplicateur le gain est théoriquement égal à 1).

Par inversion de l'intégrale de Fourier on a

$$B_\sigma(\sigma) = \int_0^{+\infty} \mathcal{J}(\Delta) \cos 2\pi\sigma\Delta \, d\Delta \qquad (7)$$

Cette expression montre que pour calculer $B_\sigma(\sigma)$, il faudrait connaître $\mathcal{J}(\Delta)$ sur toutes les valeurs de Δ entre 0 et ∞.

Or on ne fait varier Δ que jusqu'à la valeur Δ_M maximale, ce qui implique qu'au lieu de la fonction $\mathcal{J}(\Delta)$ on ne connait que le produit de $\mathcal{J}(\Delta)$ par la fonction rectangle $H(\Delta)$ = rectangle $(\frac{\Delta}{2\Delta_M})$. En conséquence, lorsqu'on calcule le spectre on obtient la transformée de Fourier $F(\sigma)$ du produit $\mathcal{J}(\Delta)$.rectangle $(\frac{\Delta}{2\Delta_M})$.

Or on sait que la transformée de Fourier d'un produit de deux fonctions est égale au produit de convolution de leurs transformées, d'où :

$$F(\sigma) = B_\sigma(\sigma) * A_o(\sigma) \qquad (8)$$

où $$A_o(\sigma) = \frac{\sin 2\pi\sigma\Delta_M}{2\pi\sigma\Delta_M}$$

La largeur de la fonction $A_o(\sigma)$: $d\sigma = \frac{1}{2\Delta_M}$

Cependant la fonction $A_o(\sigma)$ convient mal en raison de ses extrema secondaires importants. On peut choisir $H(\Delta)$ de manière convenable en prenant par exemple $H(\Delta) = \text{triangle}(\frac{\Delta}{\Delta_M})$

où triangle (x) est une fonction définie par

triangle (x) = 0 si $|x|>1$

triangle (x) = $1-|x|$ si $|x|<1$

(L'apodisation pourra se faire en affaiblissant le signal fourni par le détecteur dans un rapport variable avec la différence de marche).

Si $H(\Delta) = \text{triangle}(\frac{\Delta}{\Delta_M})$ alors $A_o(\sigma) = \frac{\sin^2 \pi\sigma\Delta_M}{(\pi\sigma\Delta_M)^2}$

dont la largeur s'écrit $d\sigma = \frac{1}{\Delta_M}$ et la résolvance est $R_o = \sigma\Delta_M$

I-2 Spectrométrie par transformation de Fourier du laboratoire

L'interféromètre construit au laboratoire de Spectronomie Moléculaire est destiné à l'étude des gaz moléculaires dans l'infrarouge moyen (6 µm-20 µm). La différence de marche maximum permet de disposer d'une fonction d'appareil dont la largeur à mi-hauteur est du même ordre de grandeur que la largeur Doppler-Fizeau des raies des gaz étudiés.

Vers 12 µm, les largeurs Doppler-Fizeau sont de l'ordre de 0,001 cm^{-1}, en conséquence la différence de marche optimum sera de 1000 cm ou 10 mètres. En fait on peut créer une différence de marche maximum de 13,5 mètres.

On pourra séparer deux éléments spectraux voisins si les ordres d'interférence des deux éléments diffèrent de 1 pour la différence de marche maximum (critère de Rayleigh)

$$d\sigma = \frac{1}{\Delta_M} \qquad (9)$$

le pouvoir de résolution est $R_o = \sigma\Delta_M = \frac{\Delta_{max}}{\lambda}$

D'autre part le spectre étudié étant borné, l'interférogramme est lui aussi borné et d'après le théorème de Shannon il suffit d'échantillonner l'interférogramme avec un pas dépendant de la largeur du spectre étudié pour pouvoir reconstituer celui-ci sans ambiguité [20]

I-2-1 Mesure de la différence de marche et asservissement du chariot mobile [10]

On mesure la différence de marche à l'aide d'un laser hélium-néon stabilisé au creux du Lamb-dip. Dans l'interféromètre, les deux faisceaux obtenus après la lame séparatrice traversent une lame quart d'onde. Les lames quart d'onde sont placées de telle manière que l'on obtienne une onde circulaire droite et une onde circulaire gauche. Au passage de la mélangeuse, les deux ondes se recombinent pour former une onde rectiligne dont l'azimut dépend de la différence de marche entre les deux ondes circulaires. L'onde rectiligne est alors divisée en deux, puis dirigée sur deux photomultiplicateurs à travers des polariseurs à $\frac{\pi}{4} = \frac{\lambda_r}{16}$.

Les signaux à la sortie des photomultiplicateurs s'écrivent :

$$S_1 = I_1 \cos^2 \frac{\pi\Delta}{\lambda_r} \qquad (10)$$

$$S_2 = I_2 \cos^2 \frac{\pi}{\lambda_r}\left(\Delta + \frac{\lambda_r}{2}\right) \qquad (11)$$

où λ_r est la longueur d'onde du laser de référence. Le controle de ces signaux est fait à tout instant par un oscilloscope. On utilisera ces signaux pour élaborer un signal d'erreur qui commandera l'asservissement du chariot portant l'oeil de chat mobile. Pour cela il faut que S_1 et S_2 soient proportionnelles à $\Delta-\Delta_S$ où Δ est la différence de marche effective et Δ_S la différence de marche souhaitée. On module alors la différence de marche à la fréquence f_o = 100 kHz. C'est à dire que l'on applique une tension sinusoïdale à une céramique piezzo-électrique supportant un miroir de l'un des oeils de chat.

Pour que l'atténuation du signal infrarouge soit négligeable, on choisit une amplitude de modulation très faible de l'ordre du dixième de la longueur d'onde de référence.
On a alors :

$$\Delta = \Delta_o + \varepsilon\lambda_r \sin \omega t \qquad \text{avec } \omega = 2\pi f_o \qquad (12)$$

Δ_o : différence de marche moyenne et $\varepsilon << 1$.
Le signal de sortie des photomultiplicateurs contient des termes aux fréquences $0, f_o, 2f_o$, etc. Après un passage dans un filtre à large bande centré sur f_o et à très forte pente pour donner une forte atténuation à $2f_o$,

$$S_1' = \pi I_1 \varepsilon \sin(2\pi\Delta_o\sigma_r x) \sin \omega t \qquad (13)$$

$$S_2' = \pi I_2 \varepsilon \cos(2\pi\Delta_o\sigma_r x) \sin \omega t \qquad (14)$$

A l'aide d'un atténuateur on réalise l'égalité des deux amplitudes ($I_1=I_2=I$). Les signaux sont alors mis en quadrature à la fréquence f_o, on obtient :

$$S_1'' = \pi I\varepsilon \sin(2\pi\sigma_r\Delta_o x) \sin\omega t \qquad (15)$$

$$S_2'' = \pi I\varepsilon \cos(2\pi\sigma_r\Delta_o x) \cos\omega t \qquad (16)$$

A l'aide d'un sommateur on a

$$S = S_1'' + S_2'' = \pi I\varepsilon \cos 2\pi(f_o t - \sigma_r\Delta_o) \qquad (17)$$

Ce signal a une amplitude constante (indépendante de Δ_o) mais sa phase est proportionnelle à Δ_o. Si le chariot se déplace à vitesse constante ($\Delta_o=vt$) alors S devient

$$S = \pi I\varepsilon \cos 2\pi t(f_o - \sigma_r v)$$

on constate que cette fréquence croît ou décroît suivant le sens de déplacement. Le signal passe ensuite par un circuit logique qui transmet une impulsion à chaque passage à zéro. Ces impulsions sont admises dans un additionneur 8 bits, qui effectue la soustraction avec des impulsions synchrones du signal 100 kHz. La sortie de l'additionneur représente le signal d'erreur sous forme digitale. La plage de réponse est $\pm 256 \frac{\lambda_r}{8}$ soit ± 20 µm. Un convertisseur analogique fournit le signal d'erreur qui est alors envoyé sur la boucle d'asservissement. Les variations de la différence de marche sont obtenues en envoyant des impulsions sur l'additionneur, chaque impulsion entrainant une augmentation de celle-ci de $\frac{\lambda_r}{8}$. On peut ainsi générer toute forme de variation de la différence de marche en

régime asservi.

I-2-2 Obtention d'un interférogramme [18]

Lorsque l'on s'éloigne de la différence de marche nulle, le signal devient rapidement petit devant un signal constant continu. Pour pallie ce problème, on va effectuer une modulation interne de la différence de marche Pour chaque valeur Δ_A de la différence de marche celle-ci subit une modulatior en créneaux. La différence de marche passant successivement de la valeur $\Delta_A-\Delta/$ à $\Delta_A+\Delta/2$ la valeur de $\mathcal{J}(\Delta_A)$ sera obtenue par détection synchrone du signal alternatif donné par le recepteur. L'intensité s'écrit :

$$\mathcal{J}(\Delta_A) = \int_0^{+\infty} B_\sigma(\sigma)\left[\cos 2\pi\sigma(\Delta_A+\frac{\Delta_M}{2}) - \cos 2\pi\sigma(\Delta_A-\frac{\Delta_M}{2})\right] d\sigma \qquad (18)$$

$$\text{donc} \quad \mathcal{J}(\Delta_A) = \int_0^{+\infty} B_\sigma(\sigma) \sin \pi\sigma\Delta_M \cdot \sin 2\pi\sigma\Delta_A \, d\sigma \qquad (19)$$

On obtient $B_\sigma(\sigma)$ par transformée de Fourier inverse si $\sigma\Delta_M = \frac{1}{2}$ c'est à dire si $\Delta_M = \frac{1}{2\sigma}$. On obtient ainsi l'interférogramme dérivé. Par cette méthode le fond continu est supprimé.

I-2-3 Calcul du spectre

On a vu que l'on obtenait

$$F(\sigma) = \int_0^{+\infty} \mathcal{J}(\Delta)\, H(\Delta) \cos 2\pi\sigma\Delta \, d\Delta \qquad (20)$$

L'intégrale sera remplacée par une somme portant sur un nombre discret de valeurs relevées sur l'interférogramme. Or d'après le théorème de Shannon, toutes les informations concernant une fonction dont le spectre a une étendue limitée ℓ sont contenues dans une infinité dénombrable de valeurs discrètes de cette fonction distantes de $1/\ell$. On relève donc sur l'interférogramme les valeurs $\Delta_o, \Delta_1, \ldots, \Delta_n$ correspondant à des abscisses équidistantes $0, h, \ldots, nh$ avec $nh = \Delta_{max}$ et on calcule :

$$F'(\sigma) = h\left(\frac{1}{2} H_o \Delta_o + \ldots + H_n \Delta_n\right) \qquad (21)$$

où les H_i sont les valeurs prises par $H(\Delta)$ pour les valeurs correspondantes de Δ. Le spectre est calculé par une transformation de Fourier numérique. Il est ensuite apodisé par convolution avec la transformée de Fourier de la fonction d'apodisation et est disponible pour être tracé avec un échantillonnage cinq fois plus serré que l'échantillonnage de départ.

II - TRANSFORMATION DE FOURIER

II - 1 Représentation d'une fonction périodique par une série de Fourier dans l'intervalle $]-\infty,+\infty[$

Ayant une fonction f(t),on peut la représenter par une série de Fourier dans l'intervalle fini (t_o,t_o+T). En dehors de cet intervalle, la fonction et sa représentation en séries de Fourier ne sont pas égales. Cependant, on montre que la représentation par des séries de Fourier est valable dans l'intervalle entier $]-\infty,+\infty[$.

Ceci peut être vu aisément en considérant une fonction f(t) et sa représentation par des séries de Fourier sous forme exponentielle dans l'intervalle (t_o,t_o+T). On peut écrire

$$f(t) = \sum_{n=-\infty}^{+\infty} F_n \, e^{jn\omega_o t} \qquad (22)$$

Cette égalité est vraie quand $t_o<t<t_o+T$. Les deux membres de l'équation (22) ne sont pas égaux en dehors de cet intervalle. Il est facile de voir, cependant, que le membre de droite de l'équation (22) est périodique (avec une période $T=\omega_o/2\pi$). On peut écrire :

$$e^{j\omega_o t} = e^{jn\omega_o(t+T)} \qquad \text{pour } T = \omega_o/2\pi \qquad (23)$$

Il est toutefois évident que si f(t) est périodique avec une période T, alors l'égalité (22) est vraie dans l'intervalle entier $]-\infty,+\infty[$. Ainsi pour une fonction périodique f(t),on a

$$f(t) = \sum_{n=-\infty}^{+\infty} F_n e^{jn\omega_o t} \qquad \text{pour } -\infty<t<+\infty \qquad (24)$$

$$\text{avec } F_n = \frac{1}{T} \int_{t_o}^{t_o+T} f(t)\, e^{-jn\omega_o t}\, dt \qquad . \qquad (25)$$

II - 2 Spectre complexe de Fourier

L'expansion des séries de Fourier aux fonctions périodiques permet une représentation réaliste de la fonction par ses termes en fréquences variées. Ces termes sont souvent appelés spectre de Fourier [3].
Nous avons alors deux chemins ou domaines pour représenter une fonction f(t) périodique :

- le domaine temps où f(t) est une fonction du temps
- le domaine fréquence où le spectre de f(t) est spécifié (les amplitudes de fréquences variées).

II - 3 Intégrale de Fourier

Maintenant, on peut représenter une fonction arbitraire par des termes d'exponentielles ou séries d'exponentielles dans l'intervalle $]-\infty,+\infty[$ [2].

On démontre qu'une fonction non périodique peut être représentée par une somme continue de signaux exponentiels [16]. On peut approcher ce problème de deux manières. On peut exprimer la fonction f(t) en des termes de fonctions exponentielles dans l'intervalle $]-T/2,+T/2[$ et faire tendre T à l'infini. Alternativement, on peut construire une fonction périodique de période T

telle que f(t) représente le premier cycle de ce signal périodique. Dans la limite, la période T devient infinie, alors cette fonction périodique a un seul cycle dans l'intervalle $]-\infty,+\infty[$ et est représentée par f(t). Il n'y a pas une grande différence entre ces deux méthodes.

La deuxième approche est mieux adaptée, elle permet la visualisation du processus sans altérer les fréquences du spectre équivalent.

De la relation (1) on obtient :

$$f_T(t) = \sum_{n=-\infty}^{+\infty} F_n \, e^{jn\omega_o t} \qquad (26)$$

où $\omega_o = 2\pi/T$

et $$F_n = \frac{1}{T} \int_{-T/2}^{T/2} f_T(t) \, e^{-jn\omega_o t} \, dt \qquad (27)$$

d'où les relations de Fourier :

$$F(\nu) = \int_{-\infty}^{+\infty} f(t) \, e^{-j2\pi\nu t} \, dt \qquad (28)$$

$$f(t) = \int_{-\infty}^{+\infty} F(\nu) \, e^{-j2\pi\nu t} \, dt \qquad (29)$$

II - 4 La correspondance entre la transformation de Fourier continue et la transformation de Fourier discrète

Dans le calcul des transformations de Fourier continues

par des calculateurs, il est nécessaire de la traiter sous sa forme discrète, représentée par un ensemble d'échantillons dans un intervalle fini de temps, de distances ou de fréquences.

Néanmoins, un effet néfaste est dû aux erreurs d'arrondi résultant de la quantification [4] et au calcul des quantités avec un nombre fini de digits [25]

De la relation (28) on peut déduire la transformation de Fourier discrète :

$$Y(n) = \sum_{k=0}^{N-1} X(k)\, W_N^{nk} \qquad n=0,1,\ldots,N-1 \qquad (30)$$

où $W_N = e^{-j2\pi/N}$

La notation vectorielle nous donne :

$$\vec{Y} = M.\ \vec{X} \qquad (31)$$

M étant une matrice (N,N), $\vec{X}$ vecteur d'entrée de N composantes
$\vec{Y}$ vecteur de sortie de N composantes

III - ALGORITHME DE TRANSFORMATION DE FOURIER RAPIDE (TFR)

Une méthode pour calculer la série formée par 2^m a été introduite par Yates, la généralisation à 3^m est donnée par Box. La généralisation de ces deux méthodes est donnée par Good pour calculer les séries de Fourier [16].

Dans les trois méthodes, on a essayé de réduire le nombre d'opérations complexes effectuées pendant le calcul. Pour certains problèmes, la méthode de Good [16] consistait à multiplier un vecteur N par une matrice N x N.

Cette matrice est factorisée en m matrices tel que m soit proportionnel à $\log_2 N$. Ainsi le nombre d'opérations complexes se réduirait à $N\log_2 N$ au profit de N^2.

Cooley et Tukey [12] se sont intéressés à l'élaboration d'algorithmes pour le traitement des séries complexes de Fourier. Ils ont mis au point une méthode de calcul du nombre d'opérations complexes et l'utilité de choisir pour le nombre de points complexes N, une puissance de 2.

Cet algorithme est général dans le sens où il n'impose pas de base de décomposition. Cependant il existe des algorithmes traitant séparément chaque base.

Ainsi, tous les algorithmes traitant la transformation de Fourier discrète sont dérivés de l'algorithme de Cooley et Tukey.

Singleton [24] proposa un algorithme basé sur un mixage de base de décomposition. Il montre, notamment, les conséquences de l'utilisation de certaines bases telle que la base 2 ou 4, la réduction du nombre de multiplications et d'additions et le temps mis par le calculateur pour traiter un nombre de points donné.

Par l'utilisation efficace des propriétés de la trans-

formation de Fourier discrète, Cooley, Lewis et Welch [11] ont montré l'intérêt de traiter 2N points réels par un algorithme N points complexes. En effet, la partie imaginaire ou la partie réelle devient nulle quand la fonction est réelle ou imaginaire pure et présente des propriétés de parité.

Clifford [7] donna un modèle statistique des erreurs d'arrondis dans le calcul de transformation de Fourier rapide en utilisant la représentation des réels en virgule flottante. Ces erreurs sont dûes à la représentation des coefficients de Fourier par des mots de longueur finie [7]. Ces erreurs varient en fonction de l'arithmétique utilisée [25].

La transformation de Fourier rapide est devenue une méthode de calcul de la transformation de Fourier discrète. Elle permet une réduction notable du temps de calcul par rapport à la transformation de Fourier classique [14]. Nous distinguons deux algorithmes de transformation de Fourier rapide :

- a - Algorithme Cooley-Tukey,(Annexe 1ère partie)
- b - Algorithme Sande-Tukey,

chaque algorithme est formulé pour exploiter les propriétés particulières des données à transformer ou la machine devant être utilisée pour calculer cette transformation. Les algorithmes de transformation de Fourier rapide sont a priori en nombre illimité, ceux de Cooley et Sande en constituant la base [3]. Le nombre de multiplications complexes, contenant un nombre d'additions ou de soustractions réelles est évalué à $O(\frac{N}{2}\log_2 N)$. Celui des additions ou soustractions complexes est de $O(N\log_2 N)$. Pour $N = 2^m$, le calcul s'effectue en $m = \log_2 N$ étapes et

chaque étape comprend N/2 modules élémentaires, comme le montrent les deux équations suivantes .

$$x_\ell(k) = x_{\ell-1}(k) + W_N^{nk}\, x_{\ell-1}(k + \frac{N}{2^\ell}) \qquad (32)$$

$$x_\ell(k+\frac{N}{2^\ell}) = x_{\ell-1}(k) - W_N^{nk}\, x_{\ell-1}(k + \frac{N}{2^\ell}) \qquad (33)$$

avec $\ell = 1,m$ indice d'étape ou de récurrence

$k = 0, N-1$

$n = 0, N-1$

De $Y(n) = \sum_{k=0}^{N-1} X(k)\, W_N^{nk}$ où $W = e^{-j\frac{2\pi}{N}}$, on calcule par récurrence sur x la valeur y(n) et $y(n+\frac{N}{2})$.

On peut ignorer les (N-1) multiplications complexes

$$NB_M \simeq O(\frac{N}{2}\log_2 N) - O(N-1 + (\frac{N}{2} - 1))$$

$$\simeq O(\frac{N}{2}\log_2 N) - O(\frac{3N}{2} + 2) \qquad (34)$$

Pour obtenir le nombre de multiplications réelles, on peut développer les équations (14). On aura alors

$$W_N^{nk} = C(k) + i\, S(k) \qquad (35)$$

$$x_{\ell-1}(k) = a_{\ell-1}(k) + i b_{\ell-1}(k) \qquad (36)$$

$$x_{\ell-1}(k + \frac{N}{2^\ell}) = a_{\ell-1}(k + \frac{N}{2^\ell}) + i b_{\ell-1}(k + \frac{N}{2^\ell}) \qquad (37)$$

$\ell = 1,m$ indice d'étape

$k = 0, N-1$

d'où

$$a_\ell(k) + ib_\ell(k) = a_{\ell-1}(k) + ib_{\ell-1}(k) + c(k)\left[a_{\ell-1}(k+\frac{N}{2^\ell}) + ib_{\ell-1}(k+\frac{N}{2^\ell})\right] + iS(k)\left[a_{\ell-1}(k+\frac{N}{2^\ell}) + ib_{\ell-1}(k+\frac{N}{2^\ell})\right] \quad (38)$$

$$a_\ell(k) + ib_\ell(k) = a_{\ell-1}(k) + ib_{\ell-1}(k) + c(k)\left[a_{\ell-1}(k+\frac{N}{2^\ell})\right] - S(k)\left[b_{\ell-1}(k+\frac{N}{2^\ell})\right] + i\left[c(k)b_{\ell-1}(k+\frac{N}{2^\ell}) + S(k)a_{\ell-1}(k+\frac{N}{2^\ell})\right] \quad (39)$$

enfin

$$a_\ell(k) = a_{\ell-1}(k) + c(k)a_{\ell-1}(k+\frac{N}{2^\ell}) - S(k)b_{\ell-1}(k+\frac{N}{2^\ell}) \quad (40)$$

$$b_{\ell-1}(k) = b_{\ell-1}(k) + c(k)b_{\ell-1}(k+\frac{N}{2^\ell}) + S(k)a_{\ell-1}(k+\frac{N}{2^\ell}) \quad (41)$$

On voit qu'il faut multiplier par 4 le nombre de multiplications complexes pour aboutir aux nombres de multiplications réelles. On a alors :

$$NR_M \simeq O(N\log_2 N) - O(6N + 8) \quad (42)$$

Il est intéressant de remarquer que tous ces algorithmes sont basés sur un même principe qui consiste à décomposer le calcul de la transformation de Fourier discrète d'une suite de longueur N en plusieurs transformations de Fourier discrètes de longueurs plus petites de façon à exploiter la symétrie et la périodicité de l'exponentielle complexe. La mise en oeuvre de ce principe conduit à des méthodes différentes dont les performances sont comparables.

IV - ADAPTATION DE L'ALGORITHME DE COOLEY-TUCKEY A NOTRE CAS

Quand on utilise l'algorithme de Cooley-Tuckey |12| pour calculer une transformation de Fourier rapide, on suppose, en général, que la fonction est complexe. Dans ce cas tout programme qui utilise cet algorithme à besoin de deux tableaux en mémoire, l'un pour la partie réelle, l'autre pour la partie imaginaire.

On peut schématiser l'algorithme de Cooley-Tuckey par

$$I_n \xrightarrow[\text{FFT}]{} B_k \qquad \begin{matrix} I_n & \text{interférogramme} \\ B_k & \text{spectre} \end{matrix} \tag{43}$$

Si la fonction à transformer est réelle, il faut mettre à zéro le tableau associé à la partie imaginaire. Bien entendu, l'algorithme de Cooley-Tuckey |12| est applicable à toute fonction réelle, mais il conduit à un gaspillage de place mémoire et de temps de calcul.

Un algorithme dérivé |24| a été défini pour résoudre ce problème et gagner un facteur deux sur l'encombrement en mémoire centrale. D'où la transformation du schéma de Cooley-Tuckey en un nouveau schéma :

$$\underset{\substack{\text{réel} \\ 2N \text{ points}}}{I_n} \longrightarrow \underset{\substack{\text{complexe} \\ N \text{ points}}}{E_n} \xrightarrow{\text{FFT}} \underset{\substack{\text{complexe} \\ N \text{ points}}}{F_k} \longrightarrow \underset{\substack{\text{complexe} \\ \text{hermitique } 2N \text{ points}}}{B_k} \tag{44}$$

tel que :

$$E_n = I_{2n} + i\, I_{2n+1} \tag{45}$$

$$\begin{matrix} B_k \\ B_{N+k} \end{matrix} = \frac{F_k + F_N^*}{2} \pm W \pm k \frac{F_k - F_{N-k}^*}{2\,i} \qquad \begin{matrix} k = 0,1,2,\ldots,N-1 \\ N' = 2N \end{matrix} \tag{46}$$

Dans le cas d'une fonction réelle impaire ou paire (le raisonnement est le même pour les deux types de fonction) échantillonnée et périodique avec 2N points par période, les deux algorithmes 43 et 44 transforment la fonction soit à par r d'une séquence I_n de 2N points, soit à partir d'une séquence I_n de N valeurs partant de l'origine et complétées par N points nulles.

Pour éviter le gaspillage de la place mémoire et de temp de calcul qu'entraîne ces opérations, un nouvel algorithme est utilisé qui modifie le chemin suivi par le schéma 44.

On inverse d'abord le schéma 44, ce qui donne un nouveau schéma

$$B_k \dashrightarrow F_k \xrightarrow{FFT} E_n \dashrightarrow I_n \qquad (47)$$

hermitique complexe complexe réel

(B_k, E_k, E_n, I_n) sont reliées par les mêmes groupes de fonctions (45,46).

A partir des N points de l'interférogramme qui partent du point de symétrie on obtient un ensemble H_n de N points complexes d'une fonction hermitique et périodique tel que :

$$H_n = \frac{1}{2}\left[I_{2n+1} - I_{2n-1}\right] + i\, I_{2n} \qquad n = 0,1,2,\ldots,N \qquad (48)$$

la transformation de Fourier rapide de H_n est alors calculée par le schéma suivant :

$$B_k \rightarrow H_k \rightarrow F_n \xrightarrow{FFT} E_n \rightarrow I_n \qquad (49)$$

En remplaçant H_n par son équivalent en I_n dans la formule suivante :

$$K_k = \sum_{n=o}^{n-1} H_n\, W_{2N} + W_{2N}^{2kn} \qquad (50)$$

Sachant que B_k est la transformation de Fourier de I_n, on peut écrire B_k en fonction K_k et K_{N-k} tel que :

$$B_k = \frac{i}{2} \frac{K_k + K_{N-k}}{\sin \frac{\pi k}{N}} - K_k + K_{N-k} \quad (51) \qquad k = 0,1,2,\ldots,N-1$$

En fait, dans cette formule, B_k n'est autre que I_n et K_k est directement lié à E_n par :

$$K_k = E_{2n} + i\, E_{2n+1} \quad (52)$$

d'où un nouveau schéma :

$$B_k \dashrightarrow H_k \dashrightarrow F_k \xrightarrow{FFT} E_n \dashrightarrow K_n \dashrightarrow I_n \quad (53)$$

B_k	H_k	F_k	E_n	K_n	I_n
réelle impaire	hermitique	complexe	complexe	réelle	imaginaire impaire
N points	N points	$N/2$ points	$N/2$ points	N points	N points

Dans le schéma (43), on calcule la transformation de Fourier par l'algorithme de Cooley-Tuckey. Cet algorithme est désavantagé pour le type de fonction qu'on calcule, à savoir l'interférogramme est une fonction réelle impaire.

Les étapes schématisées par (44 à 53) montrent le nouvel algorithme qui va nous permettre de calculer la transformation de Fourier. Dans le schéma 53, on remplace B_k par I_n et les k par n dans les autres membres, on obtient alors :

$$I_n \rightarrow H_n \rightarrow F_n \xrightarrow{FFT} E_k \rightarrow K_k \rightarrow B_k \quad (54)$$

I_n	H_n	F_n	E_k	K_k	B_k
réelle impaire	hermitique	complexe	complexe	réelle	imaginaire impaire
N points	N points	$N/2$ points	$N/2$ points	N points	N points

L'interférogramme étant une fonction I_n réelle impaire partant du point de symétrie, on obtient un spectre de Fourier qui est une fonction imaginaire impaire. La symétrie du spectre nous permet de calculer une des deux parties.

V - CONVOLUTION

V - 1 Convolution

Le théorème de convolution est peut être l'un des plus puissants des théorèmes dans l'analyse spectrale. Il permet une dérivation facile d'importants résultats [2], [21]
Soient $f_1(t)$ et $f_2(t)$ on définit la convolution par la relation suivante:

$$f(t) = \int_{-\infty}^{+\infty} f_1(t)\ f_2(t-\tau)\ d\tau \qquad (55)$$

On peut la noter :

$$f(t) = f_1(t) * f_2(t) \qquad (56)$$

V - 2 Représentation graphique de la convolution

La représentation ou l'interprétation de la convolution par un graphique est très utilisée dans l'analyse des systèmes aussi bien que dans la communication.

Dans les systèmes linéaires, la convolution graphique est très adaptée dans l'analyse si l'on connait $f_1(t)$ et $f_2(t)$ graphiquement. On peut l'illustrer sur un exemple simple :

$$f(t) = f_1(t) * f_2(t) = \int_{-\infty}^{+\infty} f_1(t) f_2(t-\tau)\ d\tau \qquad (57)$$

Il suffit de remarquer que τ est la variable indépendante dans l'intégrale de convolution. Cette intégrale représente l'aire à chaque t. Le point $f(t_1)$ est obtenu en calculant la surface donnée par cette intégrale.

VI - UTILISATION DE LA TFR DANS LE CALCUL DE LA CONVOLUTION ET DE LA CORRELATION

Les applications de la TFR telles que le filtrage digital, l'analyse des spectres d'énergie, la simulation, l'analyse des systèmes, la théorie de la communication, sont basées essentiellement sur l'implémentation de la transformation de convolution ou de la corrélation en utilisant la transformation de Fourier rapide.

Par la TFR, le calcul de la convolution et de la corrélation est devenu rapide [1],[24] par rapport aux techniques habituelles utilisant l'intégrale de convolution ou de corrélation.

Si x(k) et h(k) sont respectivement deux fonctions échantillonnées de x(t), y(t) alors on peut écrire, en utilisant la relation (9) :

$$y(k) = \sum_{i=0}^{N-1} x(i)\, h(k-i) \qquad (58)$$

x(k) et h(k) sont choisies avec une période N et vérifiant :

$$N > P + Q - 1 \qquad (59)$$

$N = 2^{\gamma}$ γ entier (pour la FFT)

P et Q sont définis par le graphique suivant :

La transformée de Fourier discrète du produit de convolution est alors :

$$X(n) = \sum_{k=0}^{N-1} x(k)\ e^{-j2\pi\frac{nk}{N}} \qquad (60)$$

$$H(n) = \sum_{k=0}^{N-1} h(k)\ e^{-j2\pi\frac{nk}{N}} \qquad (61)$$

$$Y(n) = X(n)\ H(n)$$

On calcule l'inverse et on obtient :

$$y(k) = \frac{1}{N} \sum_{n=0}^{N-1} Y(n)\ e^{j2\pi\frac{nk}{N}} \qquad (62)$$

Avec l'équation (58) le temps de calcul est proportionnel à N^2 qui est le nombre de multiplications complexes. Celui de la TFR est proportionnel à $N\log_2 N$. Ainsi l'équation (62) nous donne un temps proportionnel à $3N\log_2 N$. L'utilisation de l'algorithme défini par l'équation (62), même s'il emprunte un chemin long, est plus rapide pour le calcul d'une fonction de convolution.

L'application de la TFR à la corrélation discrète est proche de la convolution par la TFR.

Si x(k) et h(k) sont des fonctions périodiques avec une période égale à N, alors on peut écrire :

$$z(k) = \sum_{i=0}^{N-1} h(i)\ x(k+i) \qquad (63)$$

la période N est choisie telle que les relations suivantes

soient satisfaites :

$$N > P + Q - 1$$
$$N = 2^{\gamma} \qquad \gamma \text{ entier (pour la FFT)} \tag{64}$$

P et Q ont la même signification que dans la convolution.
On peut calculer la corrélation discrète en utilisant l'algorithme suivant :

$$X(n) = \sum_{k=0}^{N-1} x(k)\ e^{-j2\pi\frac{nk}{N}} \tag{65}$$

$$H(n) = \sum_{k=0}^{N-1} h(k)\ e^{-j2\pi\frac{nk}{N}} \tag{66}$$

$$Z(n) = X(n)\ H^{*}(n) \qquad H^{*} \text{ conjuguée de } H \tag{67}$$

$$z(k) = \frac{1}{N} \sum_{n=0}^{N-1} Z(n)\ e^{j2\pi\frac{nk}{N}} \tag{68}$$

Les discussions pour le temps sont les mêmes que pour la convolution précédemment étudiée.
En général, pour diminuer le temps de calcul d'un certain nombre de transformations utiles, on utilise la TFR en respectant les théorèmes mis en jeu par les algorithmes.

L'ensemble de cette étude me ramène à l'écriture plus détaillée de mes algorithmes.

CONCLUSION

En spectrométrie moléculaire, on utilise souvent le résultat de transformation de Fourier ou de son inverse. Ce fait est lié à l'outil même de l'analyse des spectres, l'interféromètre de Michelson qui fournit la transformée du spectre, ensuite, par une transformation de Fourier inverse, on reconstitue le spectre pour son étude.

Il est extrêmement important de noter que les propriétés spectrales de l'interférogramme, donc sa T.F., rassemblées dans ce chapitre sont celles de modèles mathématiques de grandeurs physiques. L'expérimentateur devra prendre un grand soin dans la mesure de ces grandeurs physiques pour pouvoir bénéficier des propriétés de la T.F.. Par exemple, pour tous les échantillons de l'interférogramme, la ddm devra être connue avec une incertitude très petite devant la longueur d'onde moyenne du spectre étudié et sa valeur mise sous forme digitale avec une précision suffisante pour ne pas introduire de défauts de quantification. Il peut arriver que le choix de tel modèle conduise à une situation paradoxale à laquelle il n'est pas toujours facile d'apporter une solution. Cependant la simplification apportée dans les calculs peut être telle que cette situation soit acceptée, au moins comme une approximation de la situation réelle.

Une difficulté est inhérente à la théorie du signal qui tient à la nature physique des espaces utilisés (longueur, longueur d'onde). La valeur d'une T.F. en un point donné est déterminée par la totalité (ou presque totalité) des valeurs de la fonction (à transformer) dans l'espace image. La T.F. d'un signal

physique est clairement définie dans les expériences en "temps différé", par exemple avec enregistrement.

On a introduit dans ce chapitre des algorithmes permettant le calcul rapide pour améliorer le traitement des spectres en spectrométrie moléculaire.

CHAPITRE II

INTRODUCTION

L'état de l'art dans l'intégration de la TFR est limité à un nombre de points petit (quelques centaines de points). Cette intégration est liée à la logique utilisée pour augmenter la vitesse de calcul et le nombre de points à transformer. Actuellement, certains circuits spécialisés, intégrant des multiplieurs et additionneurs peuvent calculer des TFR de 1024 points complexes. D'autres circuits disposent d'interpolateurs pour calculer les sinus et cosinus utilisés dans l'algorithme de la TFR. Au delà de 1024 points, on fait appel à des composants modulaires. On modularise ainsi la TFR pour la paralléliser et permettre son intégration et son implémentation sur des circuits à très haute intégration.

Cette vision est applicable à toutes les fonctions complexes ou réelles pour lesquelles il existe une transformation de Fourier. Actuellement, on tente d'améliorer la décomposition en modules pour réduire les difficultés d'intégration et d'implémentation. Maintenant, on essaie d'utiliser les circuits à très large échelle d'intégration [13] (VLSI en nomination anglaise , very large scale integration) qui sont adaptés aux algorithmes numériques modulaires et qui peuvent augmenter la vitesse du calcul de la TFR.

La majorité des circuits LSI et VLSI qui peuvent calculer une TFR sont limités en vitesse et en nombre de points à transformer. Le choix de la logique d'intégration dépend de la vitesse d'échantillonnage du signal analogique à transformer et du temps de réponse exigé de la TFR (figure II-1).

La figure II-1 montre le choix de la logique à utiliser en fonction de la largeur de la bande passante du signal analogique et de la complexité algorithmique (nombre de multiplications). [8]

La partie hachurée donne une idée de la position de nos travaux effectués sur la TFR . La flèche indique l'évolution rapide des composants modulaires pour l'intégration de la TFR.

Le critère de rapidité nécessite une exploitation optimale du parallèlisme de la transformation de Fourier rapide. D'une façon générale, on peut décomposer le problème de la parallèlisation de la transformation de Fourier en deux parties

- parallèlisme local
- parallèlisme global [15]

Dans le parallélisme de type global, la transformation de Fourier rapide est divisée en modules pouvant être exécutable en parallèle. Dans le parallélisme de type local, le module peut être décomposé en sous expressions exécutables en parallèle

Ce chapitre est en fait considéré comme un avant propos à des techniques nouvelles de programmation et d'intégration liées à l'algorithme défini pour une transformation de Fourier rapide d'une fonction réelle impaire.On a exclu l'approndissement théorique dans ce chapitre pour ne pas allonger les limites du temps fixé pour l'ensemble de l'application.

Différentes méthodes ont été étudiées pour parallélis l'algorithme TFR. Chacune traite un niveau de parallélisme différent. [6], [13]

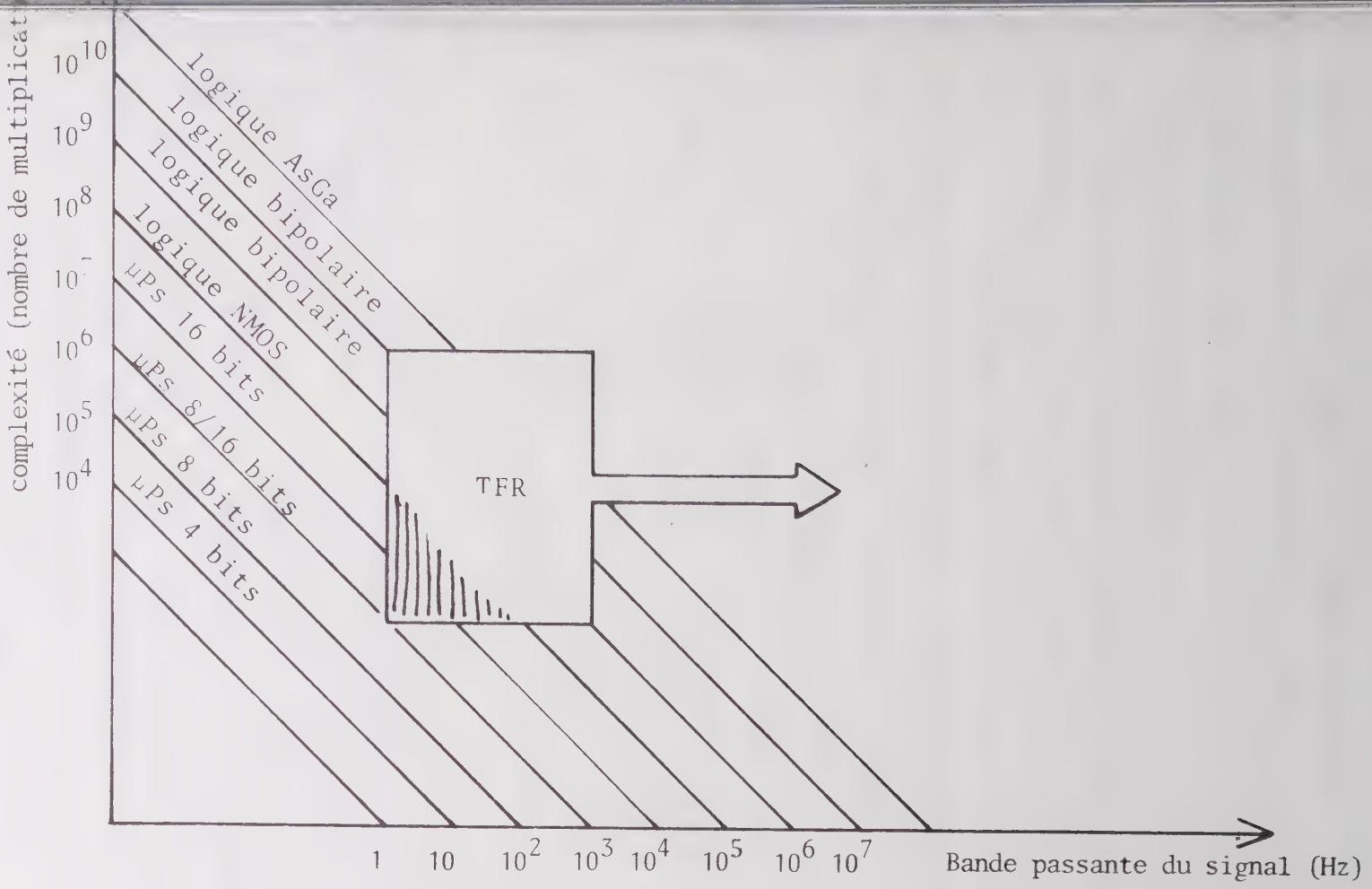

complexité (nombre de multiplicat
10^{10}
10^{9}
10^{8}
10^{6}
10^{5}
10^{4}
logique AsGa
logique bipolaire
logique bipolaire
logique NMOS
µPs 16 bits
µPs 8/16 bits
µPs 8 bits
µPs 4 bits
TFR
1
10
10^{2}
10^{3}
10^{4}
10^{5}
10^{6}
10^{7}
Bande passante du signal (Hz)

Figure II-1

Ramener le parallélisme de la TFR au niveau des opérations élémentaires permet d'augmenter le niveau du parallélisme, la taille mémoire et de diminuer le temps des transferts entre les processeurs. Il existe des méthodes de réduction de la taille mémoire ou du nombre de variables en vue d'obtenir un maximum de parallélisme dans l'algorithme [22]. Ces méthodes utilisent toujours un traitement monoprocesseur, ceci se réduit à transformer des programmes séquentiels en programmes parallèl

Localiser les fonctions et les opérateurs de base dans la TFR, leur associer des opérateurs cablés ayant des vitesses et des précisions de calcul prédéfinies par le constructeur implique l'emploi des circuits de synchronisation des opérations [10].

Ce parallèlisme reste local et ne permet pas l'utilisation de multiprocesseurs pour augmenter le nombre de points à traiter en parallèle et diminuer le temps de réponse de l'algorithme. Il apparait donc inutile de paralléliser l'ensemble des opérateurs élémentaires pour l'adapter à l'environnement multiprocesseurs. Ainsi les transferts entre processeurs seront minimum et le nombre de points maximum.

I- CIRCUIT INTEGRE SPECIALISE DANS LES TRANSFORMATIONS DE FOURIER RAPIDES

Il existe actuellement différents produits dans le marché qui concernent le calcul rapide de la transformation de Fourier. Parmi ces processeurs spécialisés, on peut remarquer ceux qui traitent un nombre fini de points et par des couplages, ils peuvent calculer autant de points que l'utilisateur désire, et ceux qui réalisent des fonctions particulières telles que les boucles de contrôle de la transformation de Fourier rapide.

Pour bien amorcer la critique de ces produits pour une utilisation éventuelle dans notre application, on ya essayer de faire le point de deux circuits typiques.

I-1 Le circuit TMC 2032 [13]

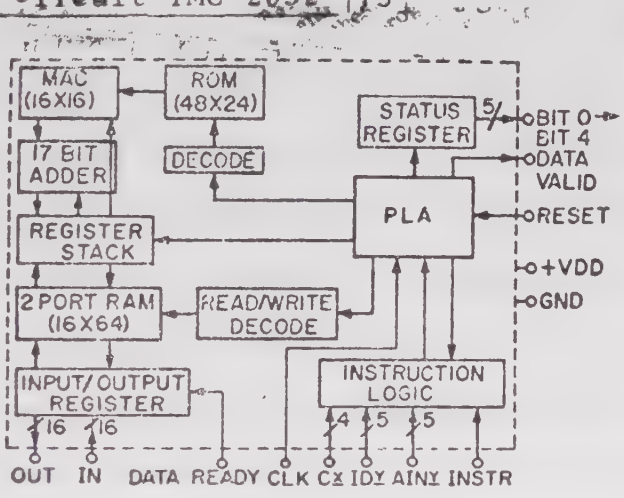


Figure II-2

Ce circuit intégré est un circuit monolithique, contenant l'algorithme de la FFT et son inverse sur 32 points complexes. Le temps mis par le circuit pour calculer une FFT d'une fonction complexe est de 47 microsecondes avec une horloge de fréquence 50 MHz. L'algorithme implanté dans ce circuit est l'algorithme de Cooley-Tuckey (decimation-in-time) en base 2.

Les entrées-sorties et les opérations arithmétiques sont calculées avec un format 16 bits, virgule fixe et complément à deux qui sont d'ailleurs communs aux processeurs spécia-

lisés rapides de traitement de signal. On remarque que les valeurs des cosinus et des sinus utilisées dans les calculs de la FFT de 32 points complexes sont stockées dans une ROM et que les calculs intermédiaires sont stockées dans des RAM.

Pour pouvoir calculer une transformation de Fourier rapide pour 1024 points, on procède ainsi :

d'abord acquérir les 1024 points de la fonction complexe, les arranger en une matrice à deux dimensions avec le format suivant :

0	1	2	3	.....	31
32	33	34	35	.....	63
.	.	.	.	.....	.
.	.	.	.	.....	.
.	.	.	.	.....	.
992	.	.	.	.....	1023

Par ailleurs, on définit M l'indexe pour les colonnes et L celui des rangées variant tous les deux de 0 à 31. Pour calculer la transformation de Fourier de 1024 points, il suffit d'utiliser le circuit TMC 2032 pour les rangées et les colonnes chaque colonne est associée à un circuit. Chaque résultat du circuit doit être multiplié par un nombre complexe qui dépend de sa position dans la matrice tel que :

$$W^{ML} \quad , \quad W = e^{-j2\pi/1024}$$

Ensuite on calcule les FFT 32 points de chaque rangée. La figur II-3 montre la mise en oeuvre d'une transformation de Fourier rapide de 1024 points d'une fonction complexe.

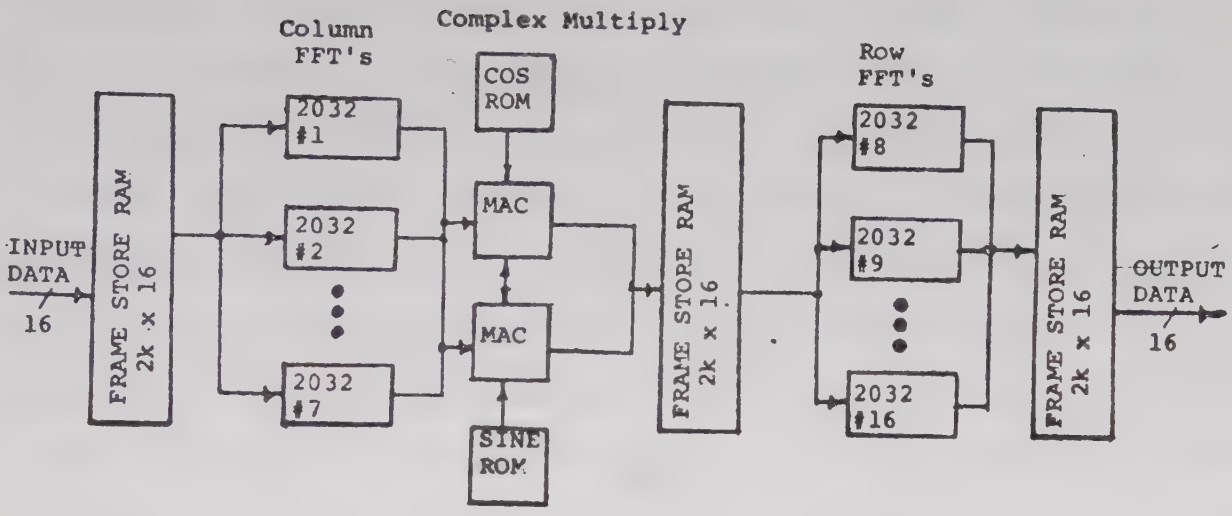


Figure II-3

On s'aperçoit alors de deux inconvénients :

1- La complexité des circuits et leur prix pour calculer la transformation de Fourier d'un très grand nombre. Il suffit d'essayer pour 4096 points.

2- Pour l'adaptation à notre cas précis, il faut lui ajouter d'autres circuits pour avoir une transformation de Fourier rapide adaptée à une fonction réelle impaire.

Il n'est pas concevable qu'à chaque changement du nombre de points on change la carte correspondante. Cette solution est très onéreuse pour une application comme la nôtre.

I-2. Le séquenceur d'adresse pour la transformation de Fourier rapide. Le Am 29540 - Figure II-4 [17]

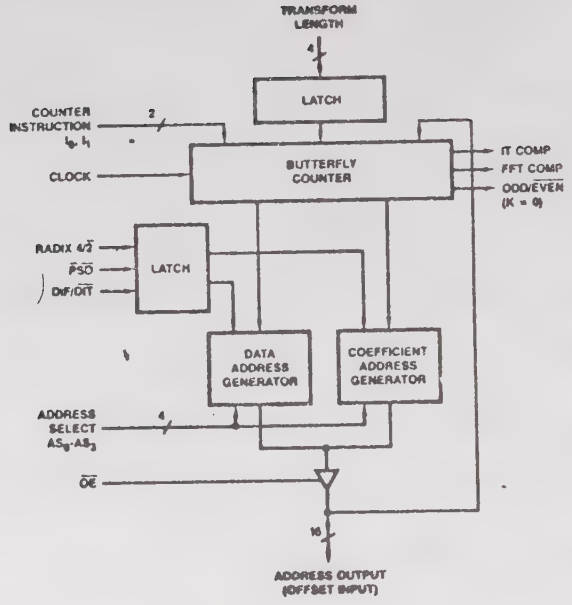

TRANSFORM LENGTH
4
LATCH
COUNTER INSTRUCTION
2
I_0, I_1
CLOCK
BUTTERFLY COUNTER
IT COMP
FFT COMP
ODD/$\overline{\text{EVEN}}$
(K = 0)
RADIX 4/$\overline{2}$
PSO
DIF/$\overline{\text{DIT}}$
LATCH
DATA ADDRESS GENERATOR
COEFFICIENT ADDRESS GENERATOR
ADDRESS SELECT
AS_0-AS_3
4
$\overline{OE}$
16
ADDRESS OUTPUT
(OFFSET INPUT)

Figure II-4

Ce circuit est un circuit intégré qui permet un adressage rapide pour la transformation de Fourier. En effet, l'adressage dans la FFT est plus compliqué que dans une transformation de Fourier classique. Certains concepteurs de circuits intégrés spécialisés dans les calculs de transformation de Fourier utilise carrément la transformation classique en utilisant des opérateurs arithmétiques rapides. Même si les multiplieurs et autres opérateurs arithmétiques apportent une solution au problème du nombre d'opérations à faire, il reste néanmoins l'autre aspect du problème, l'adressage mémoire qui pénalise sérieusement l'atout des opérateurs arithmétiques rapides. Ainsi le Am 29540 est apparu pour résoudre le problème d'adressage.

Ce séquenceur peut être utilisé avec d'autres circuits de la famille AMD comme le montre la figure II-5. En effet, il est possible d'avoir différents algorithmes au sein d'un processeur de traitement de signal, chacun d'eux est associé à un séquenceur. On peut par exemple avoir le séquenceur pour la transformation de Fourier rapide, l'autre pour l'interpolation et convolution, un troisième pour la corrélation.

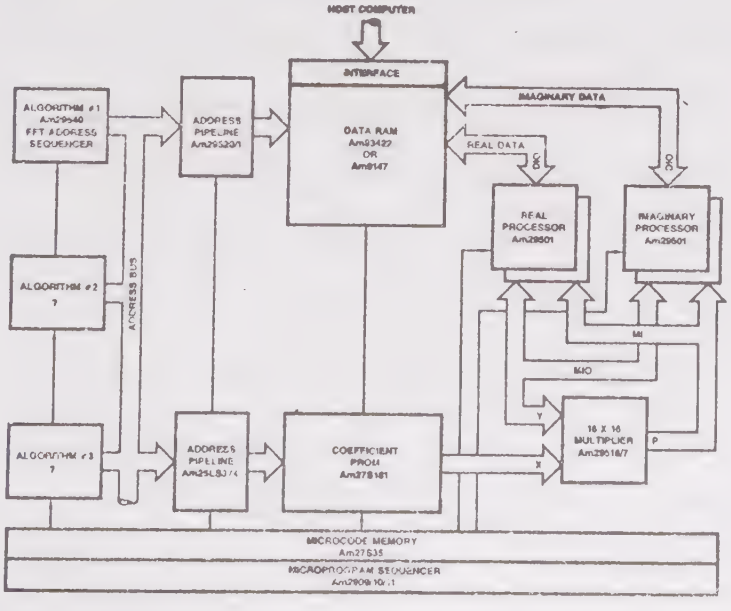


Figure II-5

Ce processeur associé à d'autres éléments tels que les opérateurs arithmétiques rapides peuvent constituer un opérateur transformation de Fourier rapide, convoluteurs rapides et autres fonctions scientifiques. Cela suppose une infrastructur de matériel électronique, d'émulateur et de programmateur pour mettre au point une carte processeur qui peut répondre à notre cas précis. Il reste à mettre au point les parties concernant le réordonnancement des points d'entrée ou de sortie, à adapter le processeur au cas d'une transformation de Fourier d'une fonction réelle impaire.

Toutefois, la complexité d'une machine finale et sa rigidité ne peuvent résoudre le problème tel qu'il a été posé. La référence à un tel produit nécessite une démarche prudente. Ce processeur associé à un système de développement approprié coûtera plus cher que l'application elle-même.

II- PARALLELISATION DE LA TRANSFORMATION DE FOURIER RAPIDE

La notion de traitement parallèle est sous-jacente à la plupart des systèmes informatiques. Que ce soit au niveau élémentaire de l'exécution de micro-instructions ou encore au niveau de la programmation de multi-processeurs et en allant jusqu'à la réalisation d'un ensemble de transactions dans un réseau.

Cependant, les propriétés d'un traitement parallèle sont assez mal définies pour les algorithmes numériques. En effet, l'aspect combinatoire des différentes situations engendrées par l'exécution simultanée d'un ensemble d'opérations, de modules ou de tâches rend très complexe l'analyse du comportement dynamique d'un programme, notamment la définition de ses états transitoires, au cours d'une exécution et met en évidence des phénomènes particuliers tels que le non-déterminisme des calculs [6], l'existence de blocage définitif [8], la réentrance d'opérations, les aspects récursifs du programme.

Toutefois, certains algorithmes présentant des symétries peuvent être transformés et découpés parallèlement. Restreindre l'étude du parallélisme dans un algorithme aux opérations élémentaires qu'il traite [15],[22], ne permet pas un découpage en modules symétriques et indépendants pouvant être traités

par des processeurs dans une configuration multibus.

Par conséquent, on recherche une symétrie dans le programme indépendante des opérations élémentaires, cette propriété de symétrie est nettement aperçue dans la TFR, convolution et autres algorithmes dépendant directement ou indirectement de l'opérateur TFR.

La mise en oeuvre de ces algorithmes dans des systèmes multiprocesseurs dépend de la souplesse de la gestion des transferts de blocs de données et des primitives d'entrée sorties utilisées par l'application. Pour atteindre l'objecti d'efficacité voulue, on peut qu'adapter ces algorithmes aux contraintes de la machine donnée. Ceci permet de contrôler la validité de ses choix d'implantations et réaliser automatiquement certaines optimisations et transformations.

II-1 Cas d'une fonction appartenant au corps des complexes

Dans cette partie, on développe la formule donnant une transformation de Fourier discrète Y(n) de $x_o(k)$, toutes les deux appartenant au corps des complexes. Ceci nous permet de rendre indépendant le calcul de chaque bloc de points de la fonction $x_o(k)$. Le calcul étant fait, on recombine les parties réelles et les parties imaginaires de telle sorte que le décalage créé au moment de la formation des fonctions $x_{io}(k_i)$. soit compensé. On utilise la propriété de linéarité de la TFR pour aboutir à la formule suivante.

On peut de

$$Y(n) = \sum_{k=0}^{N-1} x_o(k)\ e^{-j2\pi nk/N} \qquad n\varepsilon\ [0,N-1] \tag{69}$$

déduire

$$Y_1(n) = \sum_{k=0}^{2N-1} x_o(k)\ e^{-j2\pi nk/2N} \qquad n\varepsilon\ [0,2N-1] \tag{70}$$

en remplaçant N par 2N.

On peut alors décomposer $Y_1(n)$ en deux parties :

$$Y_1(n) = \sum_{k=0}^{N-1} x_o(2k)\ e^{-j2\pi nk/N} + e^{-j\pi n/N} \sum_{k=0}^{N-1} x_o(2k+1)\ e^{-j2\pi nk/N} \tag{71}$$

Le terme en $x_o(2k)$ peut être calculé par la TFR d'une fonction complexe. Il en est de même pour le terme en $x_o(2k+1)$.

On note alors :

$$TFR(x_o(2k)) = \sum_{k=0}^{N-1} x_o(2k)\ e^{-j2\pi nk/N} \qquad (72)$$

Dans le cas où on remplace N par MN, on obtient

$$Y_M(n) = \sum_{m=0}^{M-1} \sum_{k=0}^{N-1} x_o(Mk+m)\ e^{-j2\pi nk/N}\ e^{-j2\pi nm/MN} \qquad (73)$$

d'où

$$Y_M(n) = \sum_{m=0}^{M-1} TFR\ (x_o(Mk+m))\ e^{-j2\pi nm/MN} \qquad (74)$$

On peut alors répartir le traitement entre processeurs de chaque TFR.

II-1-1 Mise en forme de $x_o(k)$

La mise en forme de $x_o(k)$, appartenant au corps des complexes, en $x_{io}(k_i)$ introduit des décalages de phase. Le niveau de la recombinaison permet de les corriger.

On appelle T_{Max} la taille mémoire utile, N le nombre de points tel que $N = 2^Y$.

Le nombre de blocs est alors : $NB = \frac{N}{T_{Max}}$

on peut dans ce cas écrire :

$$x_{io}(k_i) = x_o(Mk + i) \qquad (75)$$

$$i \in [0, \frac{N}{T_{max}} - 1] \quad , \quad k \in [0, T_{max}-1]$$

$$k_i \in [0, T_{max}-1]$$

II-1-2 Recombinaison des résultats partiels

Pour recombiner les résultats de chaque TFR, on utilise la formule

$$Y_1(n) = \sum_{m=0}^{M-1} TFR(x_o(Mk+m))e^{-j2\pi nm/MN} \qquad (76)$$

On obtient en posant $TFR(x_o(Mk+m)) = XR_m^{(n)} + jXI_m^{(n)}$

et $Y_1(n) = YR(n) + jYI(n)$,

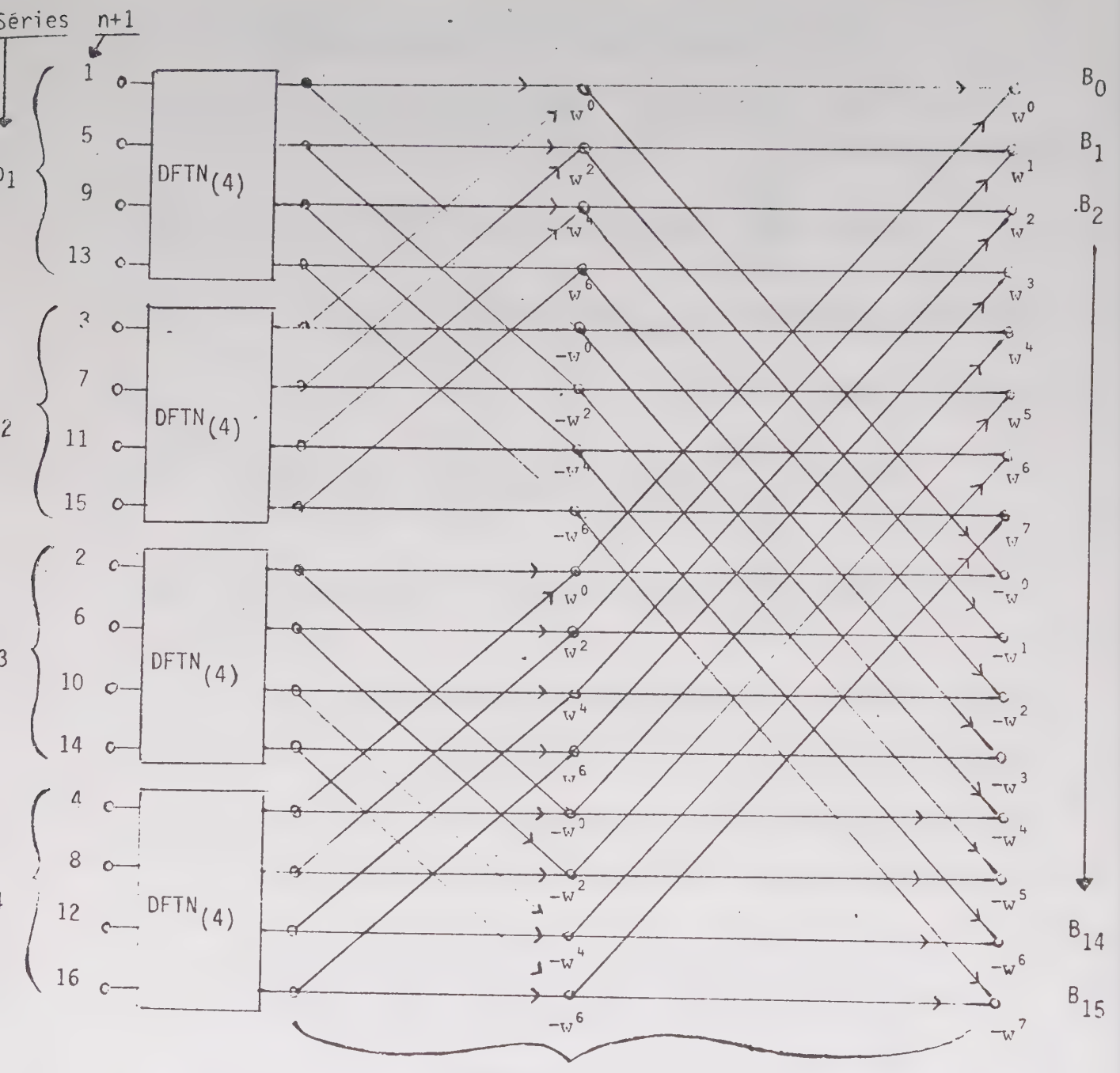
$$YR(n) = \sum_{m=0}^{M-1} (XR_m(n)\cos 2\pi\frac{nm}{MN} + XI_m(n)\sin 2\pi\frac{nm}{MN}) \qquad (77)$$

$$YI(n) = \sum_{m=0}^{M-1} (XI_m(n)\cos 2\pi\frac{nm}{MN} - XR_m(n)\sin 2\pi\frac{nm}{MN}) \qquad (78)$$

on peut faire un changement de variable tel que n=N-n pour avoir une symétrie dans les recombinaisons. La figure II-6 montre la disposition des transformations de Fourier intermédiaires et les intercombinaisons qui s'en suivent.

II-2 Cas de la fonction appartenant au corps des réels

Dans cette partie, on utilise la formule (69) pour aboutir à la formule (80). Dans ce cas la partie imaginaire est nulle. On sait que la transformée de Fourier d'une fonction réelle est une fonction complexe, on est alors amené à réserver une place mémoire à la partie imaginaire de la transformée obtenue. Pour cela, on a utilisé la partie imaginaire de la fonction réelle à transformer pour traiter deux fois plus de points que l'algorithme adapté à une fonction complexe.



(N=16, n_b=4, p=2, N_T=4)

$\{I_n\}_N \xrightarrow{DFTN_{(N)}} \{B_k\}$

$\{I_n\}_N \xrightarrow{Tri} \left\{ \begin{matrix} n_b \\ \text{séries} \\ D_j \end{matrix} \right\} \xrightarrow{n_b \times DFTN_{(N/n_b)}} \left\{ \begin{matrix} n_b \\ \text{séries} \\ \text{transfor-} \\ \text{mées} \end{matrix} \right\} \xrightarrow{p \text{ intercombinaisons}} \{B_k\}$

Figure II-6

De

$$Y(n) = \sum_{k=0}^{MN-1} x(k)\ W_1^{nk} \qquad W_1 = e^{-2j\pi/MN} \qquad (79)$$

on peut écrire

$$Y_M(n) = \sum_{m=0}^{M-1} TFR2(x_o(Mk+2m),\ x_o(Mk+2m+1).e^{-j2\pi n2m/MN} \qquad (80)$$

N et M doivent être une puissance de 2 pour que la décomposition en élément simple reste vraie.

m est défini comme le décalage des points réels de $x_o(k)$ par rapport à la valeur origine.

L'appellation TFR2 est liée à l'utilisation de l'algorithme de calcul d'une fonction complexe et aux relations de recombinaisons [5].

II-3 Représentation par des graphes de l'algorithme TFR (MN)

Ce paragraphe ne constitue pas une étude théorique par les graphes. On y montre un modèle, parmi d'autres, qui représente graphiquement l'algorithme étudié en III. Néanmoins, il est important d'utiliser quelques définitions qui serviront à construire la dualité entre les éléments de l'algorithme et ceux du graphe. Ces définitions sont les plus couramment admises ce qui n'exclut pas des formulations différentes, ni parfois les désaccords entre auteurs au sujet de certains termes. On utilisera les définitions satisfaisant les graphes orientés.

Nous représenterons la relation (80) par un graphe orienté incluant des sous-graphes orientés. Si l'on définit un graphe G équivalent à l'algorithme TFR (MN) par un couple d'éléments (A,U) on a alors l'ensemble A associé aux nombres de point et l'ensemble U associé à la TFR. Ainsi on a défini la dualité entre l'ensemble A de sommets et MN et l'ensemble U de relations binaires et la TFR. La relation , qui est la fonction successeur, associée à la TFR définit complétement le graphe G. En effet, pour calculer la TFR de 4N, on calcule d'abord la TFR de N et celle de 2N. Dans ce cas on représente cette succession d'opérations TFR par le graphe suivant :

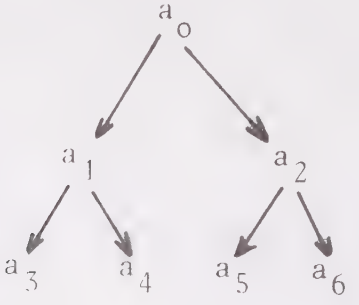


Figure II-7 Graphe G

On voit alors que les successeurs de a_o sont, par Γ ,a_1 et a_2. Ceux de a_1 et a_2 sont (a_3, a_4) et (a_5, a_6) respectivement. Pour le cas général on a :

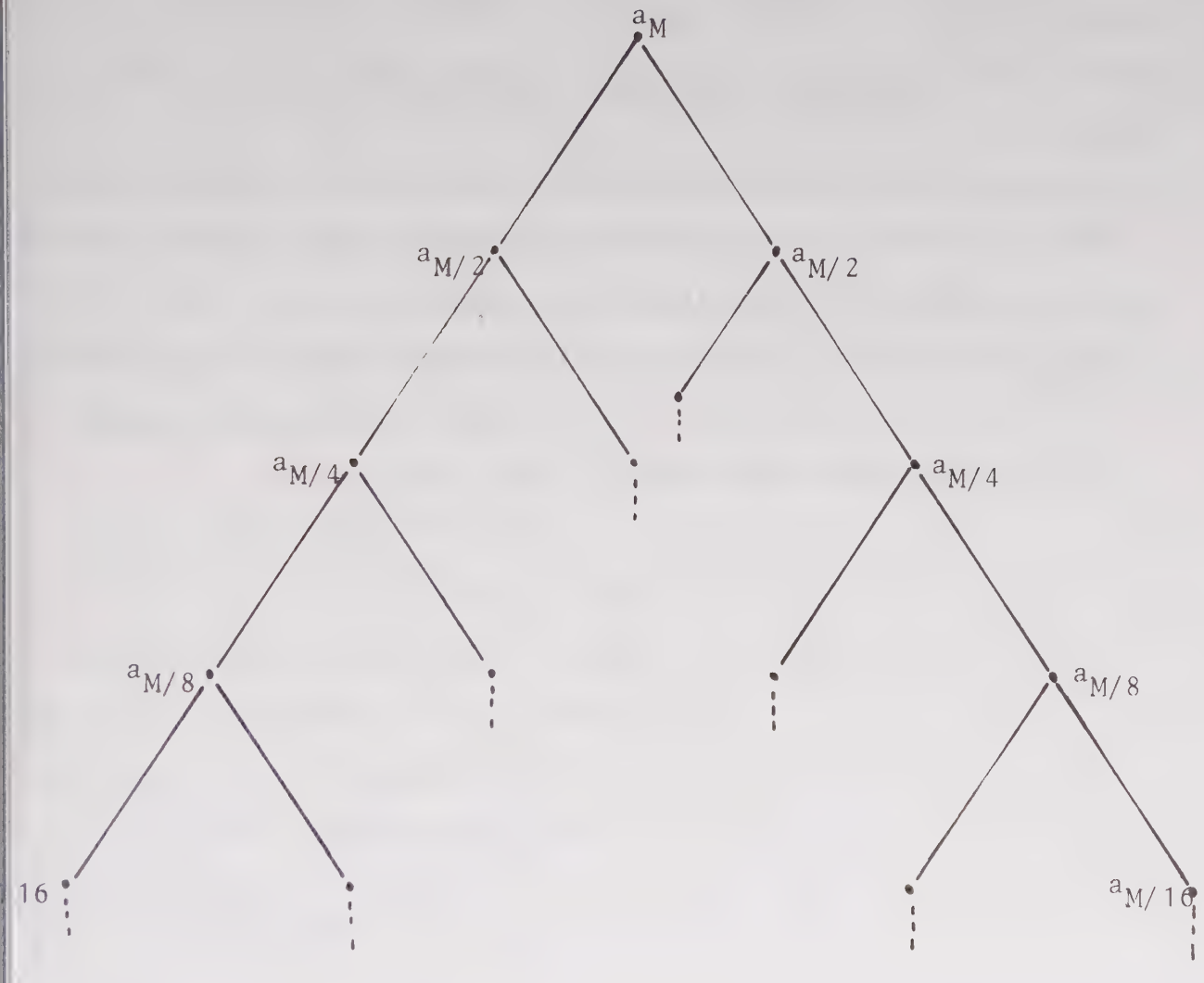


Figure II- 8 Graphe G_1

Par construction, le graphe G_1 est orienté. Il est antisymétrique. En effet pour calculer une TFR de 4N, on calcule celle de 2N. Pour calculer celle de 2N, on calcule celle de N et ainsi de suite. Pour calculer la TFR de N, on ne peut pas calculer celle de 2N pour déduire celle de N.

II-4 Estimation des temps théoriques de calcul.

L'algorithme de la transformation de Fourier rapide dans le cas des multiprocesseurs dépend du système dans lequel il est implémenté. L'estimation du temps ou de la complexité algorithmique est liée à la notion espace-temps. Si on ajoute les temps d'ordonnancement et des intercombinaisons, on peut écrire une formule approchée du temps de calcul :

$$T_C \simeq C\left(\sum_i t_{c_i}\right)/P + B$$

t_{c_i} étant le temps de calcul de la $FFT2N_i$ associé au processeur P_I, P le nombre de processeurs du système.

B temps d'ordonnancement et d'intercombinaisons

(81)

III - ALGORITHME PARALLELE POUR UN GRAND NOMBRE DE MESURES

La capacité mémoire RAM est limitée à 64 K octets partagée entre le système ISIS et moniteur, le code du programme exécutable, les données numériques et la pile. Pour augmenter la capacité de traitement, on est ramené, et ceci est valable dans les gros systèmes [8], à traiter bloc par bloc, en reliant les blocs les uns aux autres. Certes, c'est toujours un algorithme séquentiel mais il peut être parallélisé pour l'adapter à d'autres configurations de systèmes.

Pour calculer la transformation de Fourier rapide adaptée à notre application, on doit se servir des algorithmes proposés au chapitre I.

- De la relation

$$H_n = \frac{1}{2}(I_{2n+1} - I_{2n-1}) + i\, I_{2n}$$

pour $n=0,1,2,\ldots,\frac{N}{2}-1$

on déduit l'algorithme suivant

* les I_n sont répartis en blocs
* la sortie est constituée de H_{nr} réelle et H_{ni} imaginaire.

```
i ←0
j ←0
k ←0
faire a tant que i< >dernier bloc
Read (i) I_n
faire b pour I_n ←0 jusqu'à N/2 - 1
b   H_nr + iH_ni ← 1/2(I_2n+1 - I_2n-1) + i I_2n
i ←i+1
write (j,k)H_nr, H_ni
j ←j+1
```

a k ←k+1

Le programme lit un bloc i de I_n valeurs, ces valeurs sont traitées séquentiellement pour définir les valeurs H_n dont la partie réelle est progressivement stockée dans H_{nr} et la partie imaginaire dans H_{ni}. Lorsque les valeurs du bloc i ont été utilisées,les valeurs des blocs j et k sont écrites sur disquette, le processus continue jusqu'au dernier bloc. De H_n on peut calculer F_n en utilisant l'algorithme du chapitre I qui concerne les fonctions réelles impaires. On déduit alors que F_n et $F_{\frac{N}{2}-n}$, on peut écrire alors

$$F_n = f(H_n, H_{\frac{N}{2}-n}) \; ; \; F_{\frac{N}{2}-n} = f'(H_n, H_{\frac{N}{2}-n})$$

d'où l'algorithme :

i ←0 ; j ←0 * numéro du bloc

faire a tant que i, j < > derniers blocs

Read (i,j) H_{nr} H_{ni} * partie réelle et partie imaginaire, bloc i, j

Faire b pour n ←0 jusqu'à $\frac{N}{4} - 1$

$(F_{nr},F_{ni}) = f((H_{nr},H_{ni}), (H_{\frac{N}{2}-n_r}, H_{\frac{N}{2}-n_i}))$

b * pour $F_{\frac{N}{2}-n}$ les mêmes couples en utilisant f'

write (i,j) F_{nr}, F_{ni}

a i ←i+1 ; j ←j+1

La figure II-6 montre la programmation de l'algorithme pour calculer la transformation d'une fonction complexe. D'où l'algorithme

i ←0 ; j ←0 * numéro du bloc

faire a tant que (i,j) < > derniers blocs

Read (i,j) F_{nr}, F_{ni}

Appel de l'algorithme de Cooley-Tuckey

Write (i,j) E_{nr}, E_{ni}

a $i \leftarrow i+1$; $j \leftarrow j+1$

$i \leftarrow 0$; $j \leftarrow 0$

faire c tant que (i,j)< >derniers blocs

Read (i,j) E_{nr}, E_{ni}

Appel intercombinons du chapitre I

write (i,j) E_{nr}, E_{ni}

c $i \leftarrow i+1$; $j \leftarrow j+1$

L'algorithme suivant montre comment on peut obtenir de E_n les K_n valeurs réelles. On peut le programmer de la façon suivante :

* $E_n = K_{2n} + iK_{2n+1}$
* On fait correspondre à toute valeur complexe E_n deux valeurs réelles K_n

$i \leftarrow 0$; $j \leftarrow 0$

Faire a tant que (i,j)< > derniers blocs

Read (i,j) E_{nr}, E_{ni}

Faire b pour $n \leftarrow 0$ jusqu'à $\frac{N}{2} - 1$

b $(K_{2n}, K_{2n+1}) \leftarrow E_n$

Write (i,j) $K_{n/2}$, $K_{n/2}$

a $i \leftarrow i+1$; $j \leftarrow j+1$

La dernière étape de l'algorithme du chapitre I permet de calculer la tranche spectre désiré par l'utilisateur. De K_n on obtient B_n par la relation suivante :

$$B_n = f(K_n, K_{N-n}) \begin{cases} B_n = (K_n + K_{N-n})/\sin(\pi n/N) - K_n + K_{N-n} \\ B_{N-n} = (K_{N-n} + K_n)/\sin(\pi n/N) - K_{N-n} + K_n \end{cases}$$

D'où l'algorithme suivant :

```
      i ←0 ; j ←0
      faire a tant que (i,j)< > derniers blocs
      Read (i,j) K_{n/2}, K_{n/2}
      faire b pour n ←0 jusqu'à N/2 - 1
b     (B_{n/2},B_{n/2}) ←f(k_{n/2}, K_{n/2})
      Write (i,j) B_{n/2}, B_{n/2}
a     i ←i+1 ; j ←j+1
```

CONCLUSION

Nous avons donné un algorithme parallèle de la TFR dont les règles répondent aux propriétés théoriques des algorithmes existants. Le besoin de traiter un enregistrement d'un million de points par des microprocesseurs dans un temps limité pose un problème de modification de l'algorithme TFR. Cette modification a été détaillée pour une fonction appartenant au corps des réels. En outre on a montré la possibilité de paralléliser un algorithme TFR pour une fonction appartenant au corps des complexes.

On a adapté l'algorithme de la TFR au cas particulier d'une fonction réelle impaire. Il en résulte un gain de place en mémoire et une augmentation du temps machine. Cette parallélisation a apporté la souplesse de l'implémentation de la TFR dans un système monoprocesseur.

Nous pouvons remarquer que la parallélisation que nous avons décrite ne modifie pas l'ordre d'exécution des étapes de l'algorithme étudié au chapitre I.

Les progrès dans l'intégration des microprocesseurs, la mise au point des systèmes à base de circuits intégrés spécialisés, les progrès sur l'amélioration des programmes séquentiels existants permettent à la spectrométrie par transformation de Fourier rapide d'être parmi le groupe de techniques les moins consommatrices de temps machine.

Le calcul de la TFR séquentielle proprement dite n'est qu'une étape du traitement complet qu'on fait subir à l'interférogramme depuis son enregistrement jusqu'au tracé du spectre.

Une étude théorique très approfondie est nécessaire pour améliorer la programmation de l'algorithme du chapitre I. Nous avons montré qu'il existe des solutions autres que celle proposée mais mal adaptées à notre cas précis d'application.

CHAPITRE III

TRAITEMENT DU SPECTRE DE FOURIER DE L'INTERFEROGRAMME

INTRODUCTION

En spectrométrie par transformation de Fourier, l'interférogramme ne fournit pas de renseignements directement utilisables pour contrôler le déroulement de l'expérience. Ainsi un calcul rapide du spectre en totalité ou une partie est indispensable à l'utilisateur pour valider les conditions choisies pour l'enregistrement.

Un enregistrement standard dure quelques dizaines d'heures et comporte quelques centaines de milliers à quelques millions de mesures.

Au début, le nombre de points n'est pas trop important et il est possible d'effectuer une TFR sur tous les échantillons enregistrés (calcul direct). Par contre, quand le nombre devient trop grand le calcul du spectre sur une tranche suffit à contrôler l'expérience (calcul d'une tranche). Ceci est obtenu en calculant un interférogramme secondaire contenant le nombre maximum de points que l'on peut calculer.

Dans ce chapitre, on s'intéresse aussi à l'analyse du spectre total obtenu ou de la tranche calculée. Deux voies permettent d'envisager ce contrôle de l'enregistrement :

1 - Traitement à distance par un calculateur central. L'acquisition est faite par un frontal, lui-même lié au central

par une ligne de transmission téléphonique.

2 - Traitement local par un système de microordinateurs permettant une acquisition directe par une ligne à liaison série. Ceci permet d'obtenir un certain nombre de paramètres caractérisant la transformation inverse de Fourier de l'interférogramme dans un délai minimal.

A cette deuxième voie, on peut ajouter un traitement direct d'une transformation de Fourier classique [10].

Ces dernières années ont enregistré une évolution de la microinformatique en utilisant la haute integration des logiques à transistor. Ainsi les mises en oeuvre des traitements locaux se sont avérés accessibles et souples d'utilisation. La création et la commercialisation des systèmes de développement sur micro-processeur nous a permis de disposer d'un système autonome dans lequel l'implémentation d'algorithmes programmés en langage haut niveau est une réalité.

Notre travail consiste en un traitement local et la mise en oeuvre du système équivalent en interaction avec l'interféromètre de Michelson du laboratoire de Spectronomie Moléculaire.

I - CALCUL DIRECT (BASSE RESOLUTION)

Dans la mesure où l'acquisition des $2^{n'}$ points, avec $n' < n$ (n étant la puissance de 2 de la TFR) est faite, on peut alors commencer le traitement de la TFR. Pour cela on utilise l'algorithme défini au chapitre I.

Le calcul étant terminé, on dispose d'un fichier de 2^n valeurs représentant les points primaires du spectre. Une interpolation est indispensable pour pouvoir tracer ce spectre

et l'analyser.

Ainsi on peut interpoler et tracer partie par partie.

I - 1 Méthodes d'interpolation utilisées dans le tracé du spectre.

Il existe différentes méthodes d'interpolation pour approcher le spectre continu. Une des méthodes adaptées aux fonctions ayant des formes complexes est la méthode de l'interpolation par des séries de Fourier [21], [23].

Dans le calcul numérique, on peut se ramener au calcul des coefficients de Fourier. Ce calcul nécessite une infinité de termes dans la somme calculant chaque coefficient. Ce qui est impossible en calcul numérique par ordinateur. On limite alors les sommes à des calculs finis.

Le spectre à interpoler est composé d'échantillons trop espacés pour être directement tracés. On est amené à l'échantillonner avec une fréquence plus grande que celle du spectre de départ.

On utilise alors le théorème d'échantillonnage pour définir un nouveau spectre contenant un multiple de valeurs du spectre de départ. Ceci est traduit par la relation suivante :

$$y_1(\nu_n) = \sum_{k=-\infty}^{+\infty} y(\nu_k) \frac{\sin 2\pi T(\nu_n - \nu_k)}{2\pi T(\nu_n - \nu_k)} \qquad (82)$$

avec $\nu_k = k\,r$

$\nu_n = k'r'$

T étant la période de $y(\nu_k)$ filtre rectangle.

Il est donc possible, par cette relation de définir un nouveau spectre interpolé et échantillonné. $y_1(\nu_n)$ n'est pas une approximation de $y(\nu_k)$ puisque la formule (26) n'est pas une formule d'approximation.

De façon générale, on voit que le spectre $y_1(\nu_n)$ peut être défini avec un pas r' aussi petit qu'on veut. Si $r' < r$ la convolution représentée par (82) ne risque pas d'introduire les phénomènes de repliement de spectres et de recouvrement de périod qui risqueraient de distordre la partie de la fonction comprise entre $|-T, +T|$ (bande passante du filtre rectangulaire ayant pour transformée de Fourier sinc $2T(\nu_n-\nu_k)$.

Pour diminuer la durée du calcul des valeurs interpolées, on limite la somme infinie (82) à une somme finie de terme. Cette limitation introduit des erreurs sur les valeurs ainsi calculées.

On peut prendre un nouveau spectre déduit du spectre $y_2(\nu_n)$ tel que :

$$y_2(\nu_n) = \sum_{k=k_1-m}^{k_1+m} y(\nu_k)\, f(\nu_n-\nu_k) \qquad (83)$$

tel que $\nu_k = kr$, $|k_1 r - \nu| < r$

$$f(\nu_n - \nu_k) = \frac{\sin 2\pi T(\nu_n - \nu_k)}{2T(\nu_n - \nu_k)}$$

m est un paramètre choisi arbitrairement suivant les interpolations utilisées.

On peut dire alors qu'on limite la somme infinie (82) à 2 m+1 valeurs pour k=0,2 m ou que l'on tronque la somme infinie

par une fonction f (ν_n) ayant une bande passante [- ν_L,+ ν_L] avec $\nu_L = mr$.

Pour diminuer les erreurs d'interpolation liées au choix de f (ν_n), il est donc préférable de multiplier f (ν_n), filtre idéal, par une autre fonction. Cette fonction aura la propriété d'atténuer les lobes latéraux de f (ν_n).

I - 2 Algorithme de l'interpolation étudiée en I-1

On se limite à un échantillonnage 5 fois plus serré que l'échantillonnage du spectre de départ, dans le souci de simplifier l'algorithme.

De la relation (83), on peut déduire que :

$$y_2 (qr') = \sum_{k=k_1-m}^{k_1+m} y (\nu_k) f (qr' - \nu_k) \qquad (84)$$

avec $\nu_k = kr$; $m = 10$; $r' = \frac{r}{5}$

L'expression (84) devient, en remplaçant qr par $k_1 r + k_2 r'$, donc :

$$y_2 (k_1 r + k_2 r') = \sum_{k=k_1-10}^{k_1+10} y (kr) f (k_1 r + k_2 r' - kr) \qquad (85)$$

et si l'on pose $k' = k_1 - 11$ et $k = k' + n$ on aura :

$$y_2 [(k' + 11)r + k_2 r'] = \sum_{n=1}^{21} y [(k' + n)r] f [(5n - k_2 - 55)r'] \qquad (86)$$

Si $k' = k'_o$ la première valeur possible de k' et $k_2 = 0$, on obtient :

$$y_2 [(k'_o + 11)r] = \sum_{n=1}^{21} y [(k'_o + n)r] f [(5n - 55)r'] \qquad (87)$$

L'expression (87) peut être écrite d'une façon condensée sous la forme suivante :

$$y_{2(k)} = \sum_{n=1}^{21} y_i \, f_j \qquad k = 1,2,\ldots,N \qquad (88)$$

k, i et j sont liés par les relations suivantes :

$i = k_1 + n$ avec $j = 5n - k_2$

k_1 = partie entière de $\left[\frac{k-1}{5}\right]$

$k_2 = k - 1 - 5k_1$

d'où l'algorithme suivant :

1 Initialiser N (nombre de valeurs interpolées)

2 Formuler y_i $\quad i = k_1 + n$

3 Calculer f_j tel que f_j est la transformée de Fourier $\quad j = 5n - k_2$

4 Calculer la somme des produits tel que :

$$y_{2(k)} = \sum_{n=1}^{21} y_i \, f_j$$

$$k = 1,2,\ldots,N$$

I - 3 Remarques

Le but de l'interpolation dans le cas particulier des spectres de l'interféromètre, est de faire correspondre à chaque bloc de 1024 valeurs du spectre de départ cinq blocs de valeurs interpolées.

Pour que le calcul par l'algorithme d'une valeur $y_2(k_1)$ soit possible, il faut prendre en compte le retard ou le décalage introduit par l'expression (87).

Nous avons vu que N, la puissance utilisée pour le calcul de la TFR, est inférieure ou égale à M, M étant le nombre

de points à calculer. Ainsi on prendra

$$r' = \frac{1}{10Nh} \quad \text{et} \quad \nu_k = kr' \qquad 0 \leq k \leq 50$$ pour calculer

la transformation de Fourier classique du rectangle, qui est f_j.

I - 4 Filtres choisis pour l'apodisation de f_j

Pour atténuer les lobes latéraux générés par la transformée de Fourier du filtre rectangulaire, on a choisi, sans le démontrer les filtres suivants :

Pour une fonction $\begin{cases} F = 1 - \dfrac{(k-1)^2}{(Mp-1)^2} & 1 \leq k \leq Mp \\ F = 0 & k > Mp \end{cases}$

Mp est le nombre de valeurs échantillonnées dans la bande passante de F.

D'où :

- 1 - $P_k = 1 \quad \forall\, k$
- 2 - $P_k = F^2$
- 3 - $P_k = 0{,}0627 + 0{,}3221xF + 0{,}2184xF^2 + 0{,}3868xF^3$
- 4 - $P_k = 0{,}2398 + 0{,}4581xF + 0{,}2250xF^2 + 0{,}772xF^3$
- 5 - $P_k = F^{5/2}$

Suivant la fonction P_k choisie, on peut filtrer avec ou sans atténuation des lobes latéraux de sinc x. Leur pouvoir d'atténuation croît de la fonction première à la cinquième.

Pour interpoler une région du spectre, il faut tenir compte des erreurs dûes au mauvais choix du filtre, à la limitation du nombre de termes dans la sommation (82).

Il faut choisir convenablement ses filtres afin de répandre les bienfaits de l'atténuation. Cette interpolation et ces filtres ont été choisis pour tracer , avec la pré-

cision du microordinateur utilisé, des spectres satisfaisant l'expérimentateur.

II - CALCUL D'UNE TRANCHE SPECTRALE (haute résolution)

Souvent, il suffit de calculer une tranche du spectre étudié. Pour cela, un interférogramme échantillonné avec un pas k fois plus large [23] que l'échantillonnage de l'interférogramme enregistré est suffisant. Après la transformation de Fourier rapide, on obtient un spectre k fois plus étroit que le spectre origine.

On opère de la façon suivante : à partir de l'interférogramme enregistré, on calcule un interférogramme échantillonné avec un pas k fois plus grand, puis on effectue la TFR de cet interférogramme. Ceci suppose que pour un facteur de réduction donné, le découpage du spectre est imposé.

II - 1 "Réduction" du spectre

Ayant un spectre $x_o(t_n)$ primaire, on montre [14] que x(t) peut s'écrire dans le cas où le spectre est nul en dehors de la bande passante de la façon suivante :

$$x(t) = \sum_{n=-\infty}^{+\infty} x_o(t_n) \frac{\sin 2\pi\nu_1(t - t_n)}{2\pi\nu_1(t - t_n)} \qquad (89)$$

$$t_n = \frac{n}{2h}$$ h pas d'échantillonnage

Dans le cas où le spectre est à bande limitée $[-\nu_2, -\nu_1]$ et $[\nu_1, \nu_2]$ on utilise [16] la relation :

$$x(t) = \sum_{n=-\infty}^{+\infty} x_o(t) \frac{\sin[\pi\Delta\nu(t-t_n)}{\pi\Delta\nu(t-t_n)} \cos 2\pi\nu_m(t-t_n) \qquad (90)$$

$$\Delta\nu = \nu_2 - \nu_1 \quad \text{et} \quad \nu_m = \frac{\nu_1 + \nu_2}{2}$$

Pour diminuer les effets des lobes latéraux générés par sinc $(\Delta\nu(t-t_n))$ ou sinc $(2\nu_1(t-t_n))$ on les multiplie par une fonction $u_1(t_n) = \left(1-\frac{t_n^2}{T^2}\right)^2$ [16], T étant la période de $u_1(t_n)$, on obtient :

$$x_1(t_n) = \sum_{n=-\infty}^{+\infty} x_o(t)\ \text{sinc}\left[\Delta\nu\,(t-t_n)\right]\ \cos 2\pi\nu_m(t-t_n)\ \left[\left(1-\frac{t_n^2}{T^2}\right)\right]^2 \qquad (100)$$

$x_1(t_n)$ n'est pas rigoureusement identique à x(t), mais elle conserve les mêmes propriétés que celles de x(t). On peut donc réduire x(t), interférogramme de départ ou primaire en un interférogramme ayant moins de points que le premier.

Pour des raisons de filtrage [23], les bords de la tranche spectrale calculée sont déformés. Si l'on prévoit une interpolation ultérieure des valeurs primaires calculées par la TFR, on peut choisir M voisin de N, M étant le nombre de points à traiter en réduisant l'interférogramme et N la puissance de 2 utilisée pour calculer la TFR.

Dans le cas précis, où l'interférogramme est une fonction réelle impaire. Elle est connue à partir de l'origine jusqu'à M. Pour appliquer la "réduction", puisque la courbe est symétrique par rapport à l'origine, on doit tenir compte du passé de la courbe à partir de l'origine.

Il est important de remarquer que si on a le choix de la tranche à calculer, on n'a pas le choix de ses limites. En effet, pour une valeur donnée de k, le découpage en k tranches égales du spectre initial est unique.

On peut se passer du filtrage numérique pour calculer rapidement une tranche du spectre en utilisant une autre techni-

que décrite sous le nom "FFT Pruning" par John D. MARKEL [19].

II - 2 Remarques

Souvent, on fait une opération de filtrage qui permet de réduire le nombre de points del'interférogramme. Le choix de la bande passante du filtre est arbitraire. Elle dépend de la taille de la TFR à calculer. Il est évident que le calcul parallèle d'une TFR remplace le filtrage de l'interférogramme. En effet, on peut faire un calcul complet du spectre directement de l'interférogramme de départ.

En réalité le filtrage n'altère pas "l'information" contenue dans l'interférogramme. Mais le mauvais choix du filtre implique des distorsions aux bornes du spectre. Un facteur de "réduction" trop grand donne des chevauchements des périodes du spectre.

Traiter une TFR en parallèle avec un très grand nombre de points sans un filtrage préalable, diminue l'erreur d'approche du spectre théorique.

On voit donc que le filtrage est nécessaire dans le cas des petits systèmes. Leurs possibilités, tant matérielles que logicielles, ne permettent pasun traitement parallèle qui minimise les accès du disque. Ces accès du disque sont très coûteux en temps de calcul.

III - NOTIONS SUR LA RECHERCHE DE RAIES DANS LE SPECTRE

L'étude systématique des raies des spectres en local, à l'aide de la microinformatique, est devenue indispensable, pourvue que la précision des calculs satisfasse l'expérimentateur.

Le gain de temps est important vis-à-vis des traitements à distance, nécessitant des fois des jours, pour aboutir au même résultat.

Très souvent, après le calcul d'une TFR, son tracé éventuel et l'interpolation de certaines régions sont soumises à une analyse. On recherche alors les raies qui s'y trouvent. Ces raies ont une largeur à mi-hauteur, une intensité et une surface finie, dans l'approximation du premier ordre. Elles peuvent être symétriques ou dissymétriques suivant la complexité des raies trouvées. L'utilisateur choisit alors un domaine spectral, limité par deux nombres d'ondes, dans lequel il fera une recherche des raies. Ce domaine est alors interpolé et la recherche des raies y est donc limité. Il en résulte cinq étapes d'analyse d'un bloc choisi. Ces étapes seront définies plus loin sous forme d'algorithmes.

Ces étapes sont :

1 - rechercher automatiquement les raies qui se trouvent dans le bloc,
2 - calculer leur hauteur en échelle réduite
3 - ensuite la position de leur milieu et leur largeur à mi-hauteur
4 - Calculer le critère de symétrie,
5 - la surface et la largeur équivalente de la raie trouvée dans ce bloc.

Ces critères étant définis, on pourra alors dire que la raie trouvée est définie et mesurée. Sa détection est faite en comparant l'intensité de la dérivée seconde du spectre à une intensité de référence fixée ou supposée calculée par des critères définis par l'expérimentateur.

Le choix de cette intensité de référence est lié à la résolution, au rapport du signal sur le bruit et le type de raies que l'on détecte [9]. Le choix de la dérivée seconde permet l'annulation du fond continu du spectre [14].

Dans ce cas, on peut dire que si le maximum de la dérivée seconde du spectre dépasse l'intensité de référence, alors on a détecté une raie. On continue la recherche des maxima éventuels jusqu'à ce que la dérivée seconde devienne négative.

D'où l'algorithme simplifié de la détection d'une raie dans un bloc :

1	Choix du bloc du spectre résultant du calcul de la TFR à haute résolution.
2	Initialiser l'intensité de référence ou la calculer en utilisant les critères suivants : - la résolution du spectre, - le rapport signal sur bruit, - le type de raie à détecter
3	Interpolation du bloc par l'algorithme de l'interpolation I-1 en choisissant le filtre correspondant défini en I-4
4	Calcul de la dérivée seconde du spectre
5	Faire la comparaison de chaque point du bloc avec l'intensité de référence.

Les méthodes d'études de positions d'une raie détectée dans le spectre, le calcul de la largeur équivalente ont été étudiées lar

gement par les spectroscopistes utilisant la transformation de Fourier [14].

L'expérimentateur qui dispose des valeurs de ces paramètres et du tracé de la raie détecté peut alors les classer dans un fichier. Ce fichier lui permettra de disposer de toutes les données spectrales des raies du gaz étudié.

IV - CONSIDERATIONS SUR LA RECHERCHE PARALLELE DES RAIES

La recherche des raies dans le spectre est longue. Habituellement le traitement est fait par un ordinateur IBM 360/70 (500 ms par bloc).

Cette recherche est faite séquentiellement blocs par bloc. Ces bloc ont une longueur finie et définie par le résultat de la transformation de Fourier rapide (TFR).

Ainsi l'utilisation des multiprocesseurs est devenue indispensable à la recherche des raies. Chaque processeur peut prendre et traiter une tranche du spectre origine.

Comme il y a en général un grand nombre de ces blocs à étudier et que la place mémoire utile est limitée, la répartition des groupes de blocs entre processeurs augmente la rapidité de la recherche des raies. Chaque bloc lu est interpolé, traité complètement avant de passer au bloc suivant.

La longueur de chaque bloc étant arbitraire, on peut la choisir égale à 1024 points non interpolés. Après interpolation, on obtient, avec un échantillonnage cinq fois plus serré, un bloc de 5120 points.

Pour assurer la liaison entre chacun des blocs non interpolés ainsi étudiés, on fait précéder ces 5120 points des 1024 derniers points interpolés provenant du bloc précédent. S'il

s'agit du premier bloc, on le fait précéder de zéros.

Le choix du processeur pour traiter une tranche est arbitraire. Ainsi on a une indépendance de traitement de chaque tranche par un quelconque des processeurs.

V - CONSIDERATIONS GENERALES SUR LE BRUIT DANS LA METHODE DE TRAITEMENT DES SPECTRES PAR LA TFR

L'étude du bruit dans le spectre nécessite un classemer et un isolement des sources de bruit suivant le niveau de module de traitement. On peut résumer cette modularisation par la Figure III-1.

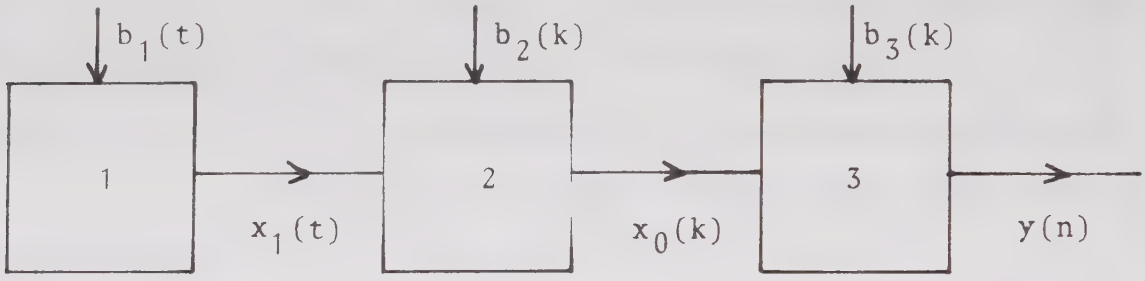


Figure III-1

1 Interféromètre

$b_1(t)$ son bruit équivalent

$x_1(t)$ interférogramme continu

2 Module d'acquisition et de conversion

$b_2(k)$ son bruit équivalent

$x_0(k)$ interférogramme échantillonné

3 Module de traitement de l'interférogramme et d'analys du spectre

$b_3(k)$ son bruit équivalent

$y(n)$ le spectre approché

Cette représentation est elle-même approchée, vu la complexité du système Interféromètre-microordinateurs.

Il est intéressant de séparer les sources de bruit créées par l'interféromètre et celles créées par les moyens de traitement de l'interférogramme [9]. Cette séparation nécessite l'indépendance des sources de bruit. En effet, il n'existe aucune relation directe entre les causes du bruit dans l'interféromètre et le traitement de l'interférogramme. La seule relation, d'ailleurs indirecte, est l'acquisition de l'interférogramme. Dans ce cas, on suppose négligeable la source équivalente au module d'acquisition et de stockage de l'interférogramme échantillonné.

Différentes études des sources de bruit provenant de l'interféromètre ont été faites dans le cadre d'une amélioration des techniques d'acquisition de l'interférogramme continu [19], [7], [25]. Parmi ces sources de bruit, la plus importante provient du bruit qui a son origine dans le détecteur de rayonnement. Dans le domaine d'utilisation de l'interféromètre, ce bruit est indépendant du signal.

D'autres types d'études ont été faites sur les sources de bruit dans le calcul et l'analyse numériques par la transformation de Fourier rapide [7]. Ces études ont montré que l'erreur moyenne et le carré moyen de l'erreur dépendent de l'arithmétique utilisée (arrondi ou troncature).

La dépendance des modules d'interpolation et de la recherche de raies avec le module TFR implique une dépendance de leurs erreurs de calcul respectives. En effet pour rechercher une raie, on utilise le résultat dela TFR, qui est lui-même entaché d'erreurs de calcul. Il convient de trouver dans l'avenir

une minimisation de l'erreur introduite par l'interpolation en augmentant la précision aux dépens de la durée de son calcul. Il n'est pas sûr que les erreurs d'un module s'ajoutent aux erreurs d'un autre module. Cependant, on démontre qu'à l'intérieur de chaque module de traitement de l'interférogramme, il y a accumulation d'erreurs d'arrondi ou de troncature [25].

Par conséquent, si un bruit est introduit au moment de la conversion de l'interférogramme analogique en un interférogramme numérique, il entachera les calculs effectués sur cet interférogramme.

L'évaluation du rapport signal sur bruit du spectre dépend du rapport signal sur bruit de l'interférogramme. Ce rapport varie comme la racine carrée de la durée totale de l'enregistrement de l'interférogramme.

VII - EVALUATION DES PERFORMANCES GLOBALES DU SYSTEME D'ACQUISITION DE MESURES.

Le rôle d'une interface d'acquisition de mesures est de prélever une information délivrée par un des capteurs de l'interface, de numériser cette information analogique, de l'amplifier éventuellement avant de la transmettre au système de traitement. En partant des capteurs jusqu'au système de traitement auquel ils sont connectés par une interface d'acquisition, le flot de mesure traverse un certain nombre de maillons dont les imperfections affectent profondément les performances globales du système. Le résultat de chaque mesure est entaché d'une certaine erreur. La précision de la mesure peut être considérablement améliorée si

l'influence des erreurs élémentaires induite par chaque maillon est identifiée et minimisée.

En bref, un interface d'acquisition de mesure constitue en soi un périphérique complet de microordinateurs, fonctionnant selon les directives dictées par les programmes implantés dans le microordinateur. La composition d'un interface d'acquisition de mesure comprend un certain nombre de modules qui sont assemblés en fonction de la précision et de la vitesse nécessaire à une bonne acquisition des mesures.

Les sources d'erreur venant perturber les mesures sur le système sont diversifiées. On note dans le système l'existence des erreurs statiques, des erreurs dynamiques. Une source d'erreur ΔS_i apportée par un module fonctionnel et exprimée en pourcentage de la pleine échelle, s'écrira : ε avec $S_i = \varepsilon \times S_{mi}$. Les erreurs statiques sont liées à la nature intrinsèque des modules qui les génèrent.

Le résultat de chaque mesure se traduit par un nombre pur représentant la grandeur électrique, délivrée par le capteur avec une certaine erreur. Cette erreur cumule le bruit qu'elle induit sur la mesure et revêt un aspect probabiliste.

En effet, on peut représenter chaque erreur par une variable aléatoire centrée sur la valeur exacte à mesurer. Elle prend au moment de la mesure une valeur inconnue puisqu'à priori on n'est maître ni des conditions climatiques, ni de l'environnement électrique, ni de la valeur des grandeurs d'influence.

La dispersion des mesures par rapport à la vraie grandeur à mesurer est caractérisée par la variance et son écart type. La variance est définie comme le moment d'ordre deux de la variable centrée, soit par définition :

$$\sigma^2 = \int_{-\alpha}^{+\infty} (v - m)^2 f(v)\, dv \qquad (101)$$

avec v variable aléatoire qui représente l'erreur

m la valeur moyenne

f(v) densité de probabilité

Cette variance représente la moyenne du carré des écarts de la variable aléatoire par rapport à m. Sa racine carré σ est appelée écart-type.

L'écart-type σ est donc bien lié directement à la précision du système. Une faible valeur de σ rend très probable les mesures proches de m, qui est par hypothèse la grandeur réelle.

Les erreurs élémentaires induites par tous les composants du système peuvent être considérées comme des variables aléatoires si on pose les deux hypothèses suivantes :

- toutes les sources d'erreur sont centrées ; elles possèdent donc une valeur moyenne nulle. Cette condition est réalisée par construction. Pratiquement, on admet - ce que l'expérience confirme d'ailleurs - que les courbes de dérivés sont symétriques par rapport à la valeur moyenne des paramètres. Il en est de même pour les autres erreurs, bruit, non-linéarités, diaphonies.

- toutes les sources d'erreur obéissent à la même loi de probabilité avec des écarts type différents.

Sous réserves des hypothèses posées, l'erreur globale considérée comme une variable aléatoire est la somme des variations aléatoires obéissant chacune à la loi normale centrée.

On démontre que la somme de variables aléatoires normales indépendantes est une variable aléatoire normale dont la moyenne est la somme des moyennes et la variance la somme des variances. Si e_i est une erreur élémentaire de densité de probabilité :

$$\frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(- e_i^2 / 2\sigma^2\right) \qquad (102)$$

l'erreur e finale a pour densité de probabilité :

$$\frac{1}{\sigma \sqrt{2\pi}} \exp\left(-e^2/2\sigma^2\right) \qquad (103)$$

la somme étant étendue à toutes les erreurs.

La grandeur mesurée $m_t + e$ où m_t est le résultat théorique de la mesure constitue elle-même une variable aléatoire de moyenne m_t et d'écart type σ.

Pour évaluer la précision, on prendra comme estimation de l'erreur 2σ, parce qu'à partir de 2σ la probabilité d'obtenir une mesure dont l'écart par rapport à m_t soit supérieur, est très faible pour la loi normale.

La connaissance mathématique de l'erreur globale sur la mesure en fonction des erreurs primaires ayant des origines diverses, nous permet d'appréhender la vraie grandeur de la mesure et ainsi d'améliorer sa précision.

CONCLUSION

Les algorithmes de calcul et d'analyse des spectres, étudiés dans ce chapitre, permettent d'obtenir un tracé du spectre avec un fichier contenant les données spectrales des raies. Ces renseignements permettent à l'utilisateur de valider l'enregistrement en cours de l'expérience et de disposer de données préliminaires pour le traitement intermédiaire.

Pour connaître certaines propriétés de la molécule constituant le gaz à étudier, il est nécessaire de rechercher dans le spectre, ses éléments spectraux ou ses raies.

L'analyse et en particulier le tracé du spectre ou de ses éléments ne peut être fait directement à partir du résultat de la transformation de Fourier rapide. Ces échantillons sont trop espacés pour l'étude visuelle et numérique de la forme des raies. Pour cela l'interpolation permet donc d'obtenir un tracé lisible.

Cette interpolation, par application d'une fenêtre sélective, introduit des distorsions aux bornes de chaque partie du spectre. Pour atténuer les méfaits de ce filtrage de ces troncatures, on emploie des filtres d'apodisation.

Sur le plan de la précision de l'ensemble des calculs, l'évaluation de l'incertitude introduite sur la mesure est liée au bruit de l'interféromètre.

CHAPITRE IV

REALISATION DU SYSTEME

INTRODUCTION

Le matériel et le logiciel de traitement des spectres sont très nombreux et diversifiés. Leur choix dépend, pour un laboratoire donné, du cahier des charges et des spécifications définis par l'utilisateur.

De ces données de départ et de l'environnement du système, on a défini un prototype de systèmes et les moyens de test pour la mise au point. On a construit des programmes à partir de ces spécifications et implanté le prototype sur le site de l'expérience : laboratoire de Spectronomie Moléculaire.

L'évolution de certaines données des spécifications pendant la durée de l'analyse a rendu plus aisé les modifications et la mise au point d'éventuelles transformations des programmes utilisateurs. Leur analyse a été faite aux chapitres II et III en tenant compte des possibilités du système adopté et des contraintes de l'environnement. Ces algorithmes ont été codés dans deux langages haut niveau : Fortran 80 et PLM 86.

Le système est couplé à l'interféromètre de Michelson, étudié au chapitre I, permettant une acquisition des mesures de l'interférogramme échantillonné et le traitement du spectre obtenu.

Dans ce chapitre, on a réalisé un système répondant au cahier des charges et aux spécifications qui lui sont propres.

Ce système, en plus du traitement des interférogrammes et de leur spectre, possède des propriétés d'extensibilité propres à résoudre la gestion et le traitement des données fournies par l'équipement de spectroscopie à haute résolution existant au Laboratoire de spectronomie moléculaire.

Ce système permet, en outre, la gestion de l'édition des parties des spectres en mode interactif. Il est décomposable en sous-systèmes, chaque sous-système est associé à une fonction définie. Ils sont reliés entre eux par des liaisons séries et parallèles qui ont la propriété de transmettre les mesures d'un sous-système à un autre sous-système.

Ces différents sous-systèmes localisés peuvent travailler en modes asynchrones. Ils présentent des particularités d'indépendance physique et logique, leur topologie respective peut évoluer avec des "conditions aux limites" inchangées. Ceci permet des substitutions des parties du système par d'autres sous-systèmes.

I - ELEMENTS DU SYSTEME (Figure IV-1)

Nous nous intéressons dans ce paragraphe aux moyens d'augmenter la fiabilité du système à réaliser. La fiabilité d'un objet est définie par la probabilité pour que cet objet remplisse pendant un temps donné des fonctions spécifiées dans certaines conditions d'exploitation. Nous allons examiner, d'une façon formelle, une des méthodes de mise au point qui tendent à accroître la probabilité de bon fonctionnement de notre système.

A l'heure actuelle, la mise au point des systèmes

tend à consommer la majeure partie du temps et de l'énergie du concepteur. Au-delà d'un certain degré de complexité, on peut dire qu'un système ne sort jamais de sa phase de mise au point, c'est-à-dire qu'il y subsiste toujours des erreurs.

L'environnement dans lequel le système doit être exploité nous a fixés le seuil de taille, de durée de réalisation, de complexité, au-dessus desquels le système proprement dit peut être qualifié d'irréalisable. Ainsi nous avons établi un modèle de système en tenant compte du problème concret. On a construit un certain nombre de programmes exécutables dans des machines données constituant notre système. Ces programmes ont satisfait les propriétés qui ont servi à définir le comportement de notre système dans les circonstances qui sont déterminées par l'environnement. Remarquons simplement que le passage du modèle de système à la réalisation et de la réalisation à la mise au point ont introduit des distorsions dues aux limitations introduites par les langages utilisés et le matériel sur lequel on a effectué l'implantation. Un système est dit valide s'il répond à ses spécifications, pour être utilisable. Il possède certaines propriétés qui influent sur son mode de réalisation et de fonctionnement.

Notre système est aisément modifiable dans la mesure où il a une éventualité de modifications explicitées et prévues dans sa construction. On a restreint les possibilités de modifications au choix d'assemblage conditionnel, et conçu l'ensemble des programmes de manière à faciliter le remplacement et l'addition de parties de programmes.

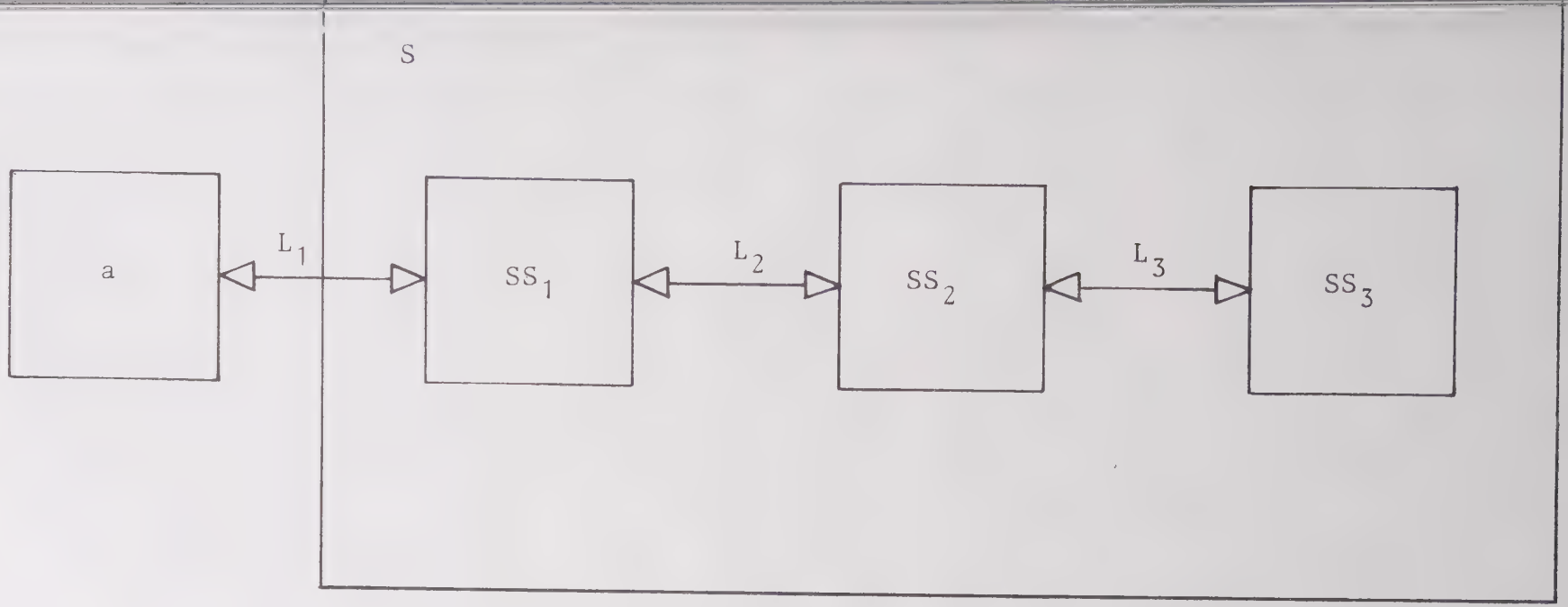
On ne connaît pas de méthode générale de décomposi-

tion des systèmes. On doit donc s'appuyer sur l'expérience, c'est-à-dire suivre dans la pratique une démarche essai-échec nouvelle décomposition.

Considérons le système S (Figure IV-1), au lieu de prouver qu'il satisfait certaines spécifications, on l'a construit pour satisfaire les spécifications données par l'utilisateur. Une solution idéale serait naturellement de construire le système S dont les spécifications tendraient à résoudre le problème général d'autonomie. L'impossibilité où l'on se trouve en général, d'exprimer formellement les spécifications rend une telle solution inexploitable dans la pratique. On tente donc, de façon plus pragmatique, de tirer parti de la liberté dont on dispose dans la conception du système S pour lui donner une structure propre à faciliter à la fois la construction du système S et la démonstration de sa validité et de sa souplesse.

Ainsi la décomposition du système S en sous-systèmes aboutit à des élèments de complexité accpetable. Le choix de cette décomposition est telle que les interactions entre sous systèmes soient les plus simples possibles.

Il est généralement admis que la complexité de la réalisation d'un système, sans chercher à définir précisément cette notion, croît beaucoup plus rapidement que le nombre d'éléments de ce système. Dans ces conditions, on a décomposer la réalisation d'un ensemble de parties plus simples et d'autre part, en l'assemblage de ces parties, on a réduit la complexité globale de l'opération. L'autonomie du système est définie par ses possibilités d'être indépendant dans un environnement donné, d'autres systèmes plus puissants.



S - Système
a - Interféromètre
SS_1 - Sous-système d'acquisition
SS_2 - Sous-système d'entrée-sortie
SS_3 - Sous-système "transformateur"

FIGURE IV - 1

Le traitement "local", souvent appelé direct, nécessi une séparation d'ensembles de fonctions. Chacune de ces fonctions est définie et localisée. On définit alors des sous-ensembles d'ensembles comme étant une partie d'un système S appelés sous-systèmes SS_1, SS_2 et SS_3. Il en résulte une décomposition du système S. Ces sous-systèmes incluent des fonctions qui leur sont propres. Cette décomposition tient compte des interfaces qui lient les sous-systèmes entre eux. Pour une fonction donnée, on associe un sous-système prédéfini et ayant des caractéristiques supoortant la fonction. On suppose que cette fonction est globalement connue et spécifiée.

Dans le cadre de notre application, on peut définir trois phases de traitement des interférogrammes qui sont :

- l'acquisition et la mise en forme des mesures de l'interféromètre.
- le traitement de l'interférogramme.
- le stockage et la visualisation des résultats.

Ainsi le système peut être décomposé en élèments simples ou sous-systèmes suivants :

- 1 - Sous-système d'acquisition ou de mise en forme des mesures
- 2 - Sous-système d'entrée-sortie et de stockage de mesures.
- 3 - Sous-système de transformation de l'interférogramme ou "Transformateur".

Cette décomposition est basée essentiellement sur les possibilités des composants matériels et logiciels composant le système global. Ainsi, le sous-système d'acquisition n'est autre qu'un moyen de gérer les modes de transferts des mesures provenant de l'interféromètre. Les éléments de la mesure sont acquis ou lus en parallèle et envoyés en série.

Le sous-système d'entrée-sortie permet la gestion globale de l'ensemble des entrées et des sorties du système. On peut alors dialoguer avec le système à travers le sous-système d'entrée-sortie, stocker les mesures sur des disquettes et faire des vérifications de mise au point. Le dernier sous-système est associé aux traitements rapides des mesures. Il transforme l'interférogramme en un spectre de Fourier.

Pour décomposer une telle application, on doit définir les interfaces caractérisant les limites de chaque sous-système. La "topologie" interne du système S est invariante. Ainsi les interfaces peuvent être banalisées. Elles sont, en général, formées d'un support physique et d'un logiciel gérant les interactions qui en résultent. Ainsi, en cas de panne, non localisée, on peut faire des tests séparés de chaque élément du système sans tenir compte des interconnections et des interactions pouvant résulter des liaisons entre eux.

Les moyens de tests sont réduits en fonction de chaque sous-système. En effet, la connaissance du sous-système et sa fonction associée, nous permet de suivre le processus en cours d'exécution sans interaction avec les autres sous-systèmes. De proche en proche, on réalise la vérification et les mises à jour dans les applications.

Dans la suite, on notera :

le système global = système S

le sous-système d'entrée-sortie = SSES ou SS2

le sous-système d'acquisition = SSAC ou SS1

le sous-système "transformateur" = SSTR ou SS3

II - LE SOUS-SYSTEME SS_1 (Figure IV-2 a)

Pour résoudre le problème d'acquisition de l'interférogramme, on a introduit une carte microprocesseur muni d'un coupleur d'isolement et d'un logiciel d'acquisition et de transfert des mesures provenant de l'interférogramme. Le coupleur et la carte microprocesseur sont deux parties distinctes, reliées par des cables à l'interférogramme. Pour acquérir la mesure et réguler la vitesse de transfert dans l'environnement où certains parasites peuvent fluctuer les échanges, on a introduit une carte microprocesseur couplée, à travers la carte d'isolement, à l'interférogramme.

Il est important d'isoler la carte microprocesseur, et donc le sous-système d'acquisition, de la haute tension provenant de l'interféromètre. Pour assurer cette fonction d'isolement, on a construit un prototype de carte coupleur.

II - 1 Le coupleur (Figure IV-2 b et c)

Cette carte permet l'isolement électrique des signaux de commande (Figure IV 2 b) et ceux des données (Figure IV 2 c) de l'interféromètre et du sous-système SS1. L'utilisation des composants optoélectroniques qui ont un pouvoir d'isolement d'un kilovolt a facilité l'interface et l'intégration du coupleur. La compatibilité et l'adaptation des signaux électriques sont assurés par des ponts intégrés de résistances.

II - 2 La carte microprocesseur

Cette carte est un microordinateur de la série SBC 80/10 (single board computer) conçue autour d'une unité centrale 8080. Cette unité centrale est un microprocesseur huit bits, il est piloté par une horloge de temps de cycle 2 µs. Un bus local qui permet les échanges entre les divers périphériques locaux de la carte. Une mémoire locale, qui est composée d'une mémoire morte de quatre kilosoctets et

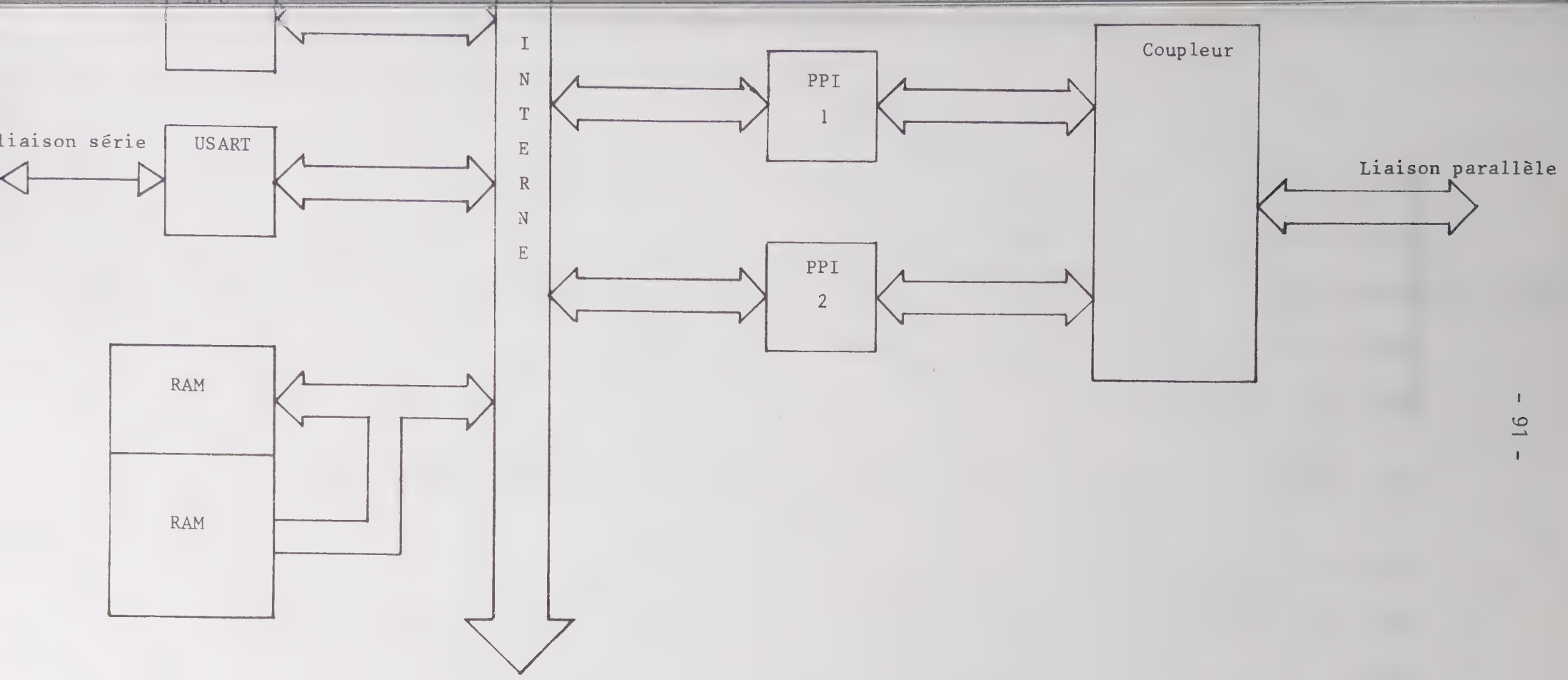
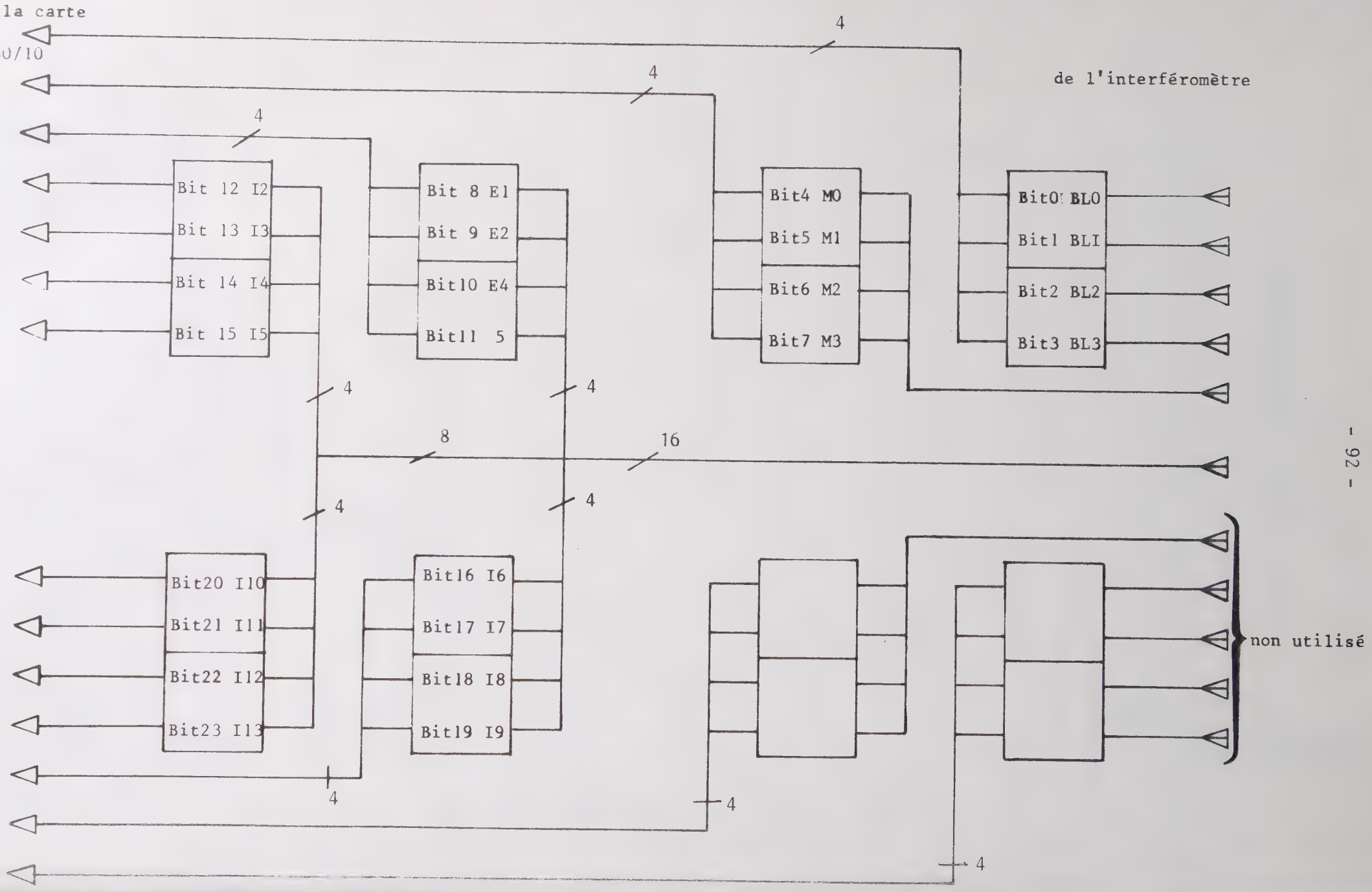

liaison série
USART
INTERNE
PPI
1
PPI
2
Coupleur
Liaison parallèle
RAM
RAM

FIGURE IV - 2a


Vers la carte
SBC 80/10
de l'interféromètre
4
4
4
Bit 12 I2
Bit 13 I3
Bit 14 I4
Bit 15 I5
Bit 8 E1
Bit 9 E2
Bit10 E4
Bit11 5
Bit4 M0
Bit5 M1
Bit6 M2
Bit7 M3
Bit0 BL0
Bit1 BLI
Bit2 BL2
Bit3 BL3
4
4
8
16
4
4
Bit20 I10
Bit21 I11
Bit22 I12
Bit23 I13
Bit16 I6
Bit17 I7
Bit18 I8
Bit19 I9
non utilisé
4
4
4

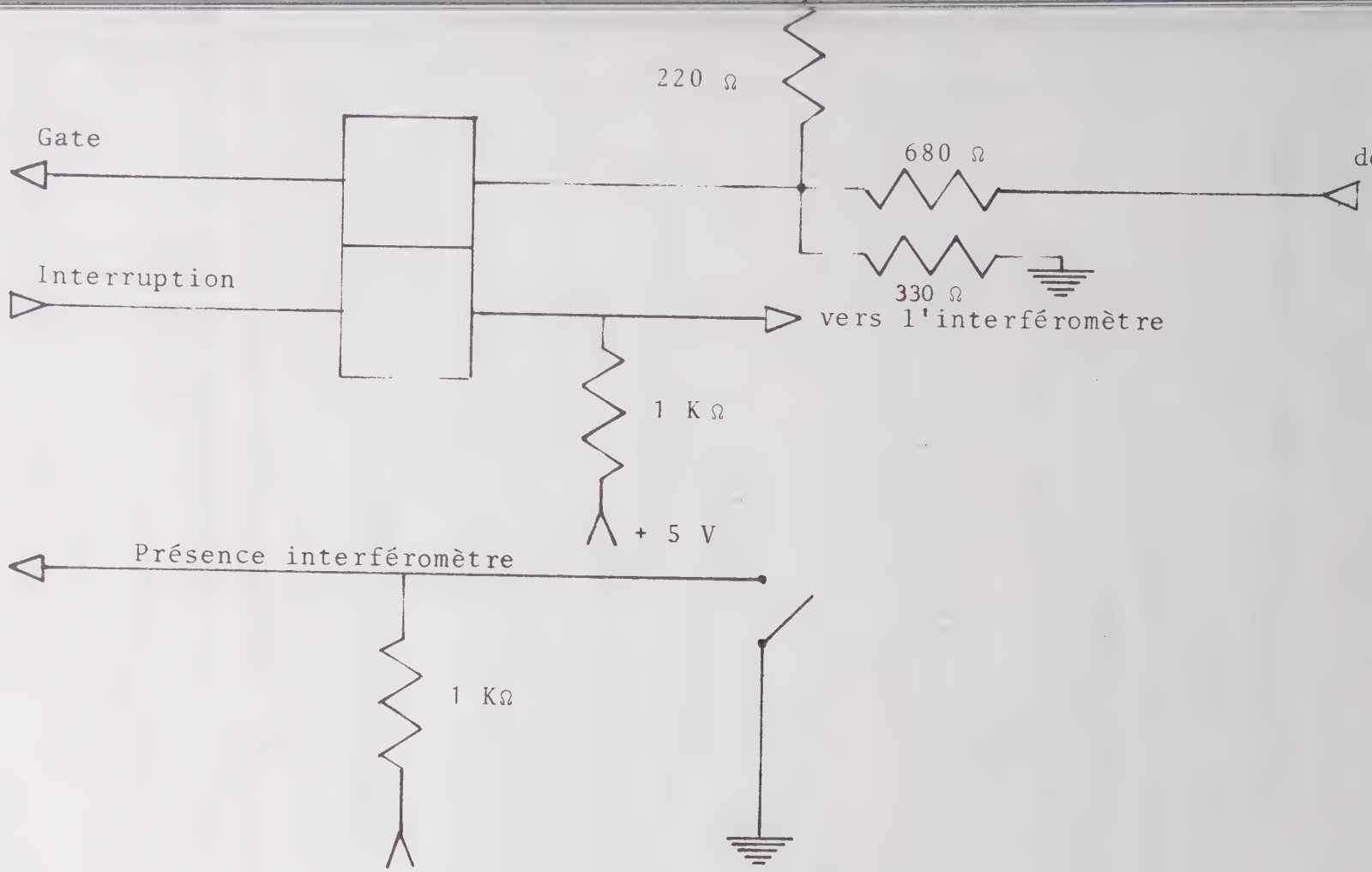


FIGURE IV - 2 c

Schéma de commande

d'une mémoire vive d'un kilooctet.

Deux périphériques de communication série et parallèle interconnectent les autres sous-systèmes avec le sous système SSES. L'interface parallèle permet de recevoir les trente deux bits, de la mesure en parallèle. Elle gère les signaux de commande.

Parmi les signaux de données, on peut citer :

- le numéro de bloc
- le numéro de mesure
- le facteur de gamme
- le signe de l'intensité
- l'intensité.

Pour gérer la validation de la mesure et la vitesse de transfert des mesures de l'interféromètre on dispose de deux signaux :

- la gate
- l'arrêt de l'interféromètre

La présence de l'interféromètre est détectée sur le signal interféromètre présent.

L'interface série envoie la mesure en sérialisant les quatre octets lus de l'interface parallèle.

II -3 Le logiciel implémenté

Pour acquérir les mesures et réguler le transfert du flot de mesures, on a mis au point une application qui permet de gérer l'acquisition et le transfert de mesures. Cette application se décompose en quatre partie :

1 - Procédure d'initialisations

2 - Procédures d'acquisition et de gestion de la liaison parallèle

3 - Procédures de gestion du tampon circulaire

4 - Procédures de gestion de la liaison série.

II -3-1 Procédures d'initialisations

Les procédures d'initialisation définissent les modes de travail de la liaison parallèle et de la liaison série. Elles permettent, en outre, l'initialisation du dialogue entre les sous-systèmes. Cette initialisation est définie par des caractères de contrôle et des signaux électriques envoyés et reçus par les sous-systèmes.

II -3-1-1 Initialisation de la liaison parallèle

La liaison parallèle est constituée de deux groupes G et G' de ports d'entrée-sortie. Chaque groupe est lui-même constitué des deux sous-groupes qu'on appellera G_1 et G_2, G_3 et G_4. Chaque groupe peut être initialisé dans l'un des trois modes suivants :

Mode 0 - entrée - sortie directe

Mode 1 - entrée - sortie strobée

Mode 2 - Bus bi-directionnel

La liaison parallèle contient six ports parallèles de huit bits qu'on peut associer au sein des sous-groupes. En effet, chaque sous-groupe est constitué par trois ports d'entrée-sortie que l'on appellera A, B et C pour G_1 et G_2 et A_1, B_1 et C_1 pour G_3 et G_4.

Le mot de contrôle de l'initialisation du groupe G est le suivant :

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
1	0	1	1	0	0	1	1

Sous-groupe G1 :

B_0=1 Port C partie basse en entrée

B_1B_2=10 Port B en entrée, mode 0

Sous-groupe G2 :

B_3=0 Port C partie haute en sortie

$B_4B_5B_6$=011 Port A en entrée mode 1 strobé

La figure IV-2 d montre la signification des bits du mot de contrôle pour l'initialisation des sous-groupes G_1 et G_2.

Pour permettre la génération de l'interruption par le sous-groupe G_1, on initialise en écrivant un deuxième mot de contrôle sur le premier groupe G. Ce mot de contrôle est généré ainsi que le montre la figure IV-2 e.

Mot de contrôle de démasquage de l'interruption

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
0	0	0	0	1	0	0	1

$B_0 = 1$ Bit utilisé pour mettre à 1 le masque

$B_1B_2B_3=001$ Permet la sélection de l'un des bits du port C

$B_6B_5B_4$ Non utilisés .

$B_7=0$ Rend actif la mise à 1 ou la remise à 0 du bit sélecté.

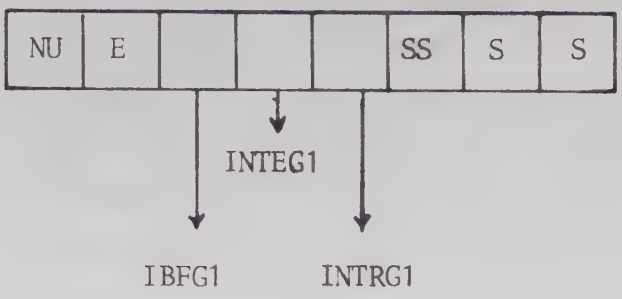
Le groupe G' est initialisé en mode 0. Ceci nous permit d'avoir les sous-groupes G_3, G_4 en entrée directe.

L'initialisation du sous-groupe G_1 en mode 1 nous a permis de disposer d'un générateur d'interruptions. L'accès à un bit précis du port C du groupe G facilite le masquage et le démasquage des interruptions.

La lecture du mot d'état du groupe G nous permet de tester la présence de l'interféromètre et celle de l'interrup-

tion. Le mot d'état est défini ainsi :

Fig. II-2d



S = en sortie

NU = Non utilisé

E = en entrée

IBFG1 = Tampon d'entrée est plein de G_1

INTEG1 = Bit de masquage et de démasquage de l'interruption

INTRG1 = Interruption de G1

B6=0 Interféromètre présent

B6=1 Interféromètre absent

B3 = Permet la génération de l'interrupteur

SS = Sans signification

L'initialisation du groupe G' en mode 0, avec les ports A_1 et B_1 en entrée et la partie haute du port C_1 en sortie, nous permet d'arrêter ou de démarrer l'interféromètre.

III-3-1-2 Initialisation de la liaison série.

Il existe deux modes de transmission de la liaison série : le mode synchrone et le mode asynchrone. Le mot de contrôle pour le mode asynchrone choisi est le suivant :

le mot de contrôle est le suivant

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
1	1	1	1	1	1	1	0

B_0B_1=01 vitesse = 9600 band

B_2B_3=11 longueur du caractère, 8 bits

B_4B_5=11 nombre de bits de stop, 2 bits

B_6B_7=11 la parité démasquée, parité paire

La lecture du mot d'état nous informe de la présence

ou de l'absence d'un caractère dans le tampon d'entrée et l'envoi ou le non-envoi du caractère contenu dans le tampon de sortie. Ce mot d'état est composé ainsi :

B_7	B_6	B_5	B_4	B_3	B_2	B_1	B_0
DSR	SYNDET	FE	OE	PE	TxEMP	RxRDY	TxRDY

Les bits testés sont les suivants :

$B_0 = 1$ Caractère envoyé, l'émetteur prêt à envoyer le caractère sur la ligne série

$B_0 = 0$ Le caractère n'est pas envoyé, l'émetteur n'est pas prêt pour recevoir le caractère de l'unité centrale.

$B_1 = 1$ Caractère reçu, le récepteur prêt à envoyer le caractère à l'unité centrale.

$B_1 = 0$ Caractère non reçu.

$B_1 = 1$ Il y a erreur de parité

$B_3 = 0$ Pas d'erreur de parité

L'écriture et la lecture est faite par des instructions du PLM 80: Input et Output.

II -3-1-3 Initialisation du dialogue du sous-système SS1 de l'interféromètre et du sous-système SS2.

La présence de l'interféromètre est nécessaire à tout traitement ultérieur. Le bit 6 du port C détecte la présence ou l'absence ee l'interféromètre.

L'algorithme d'initialisation du traitement est le suivant :

1 On arrête l'interféromètre

2 Tant que l'interféromètre n'est pas présent on

envoie un caractère de contrôle au sous-système SS2 qui signifie l'absence de l'interféromètre.

3 l'interféromètre est présent, on démarre l'acquisition

4 le sous-système SS2 présent, on envoie la mesure

5 Le sous-système SS2 n'est pas présent, on attend que le tampon soit plein, ensuite on arrête l'acquisition momentanément

6 Il y a une place dans le tampon on continue l'acquisition ensuite on revient à 4

II -3-2 Acquisition et transmission de la mesure

Le démarrage de l'interféromètre conditionne l'arrivée des mesures. La procédure de la gestion de la liaison série est activée indépendemment de l'interruption provenant de la liaison parallèle. La validation d'une mesure est faite par la transition vers le niveau bas du signal présence-mesure généré par l'interféromètre.

Ainsi après une interruption, on lit le registre d'état et on teste le bit 3 du mot d'état du groupe G. Cette précaution nous permet de discerner les évènements parasites des interruptions non définies.

Cependant la carte microprocesseur ne permet pas une utilisation plus souple des interruptions. En effet, elle ne contient pas de contrôleur d'interruptions qui gère les priorités.

A l'arrivée d'une mesure, le processus en cours d'exécution est interrompu, avec une sauvegarde de son état courant, les procédures de stockage et de lecture sont activées, une fois la mesure sauvée, on restaure l'état du processus interrompu et l'exécution continue.

La transmission est faite indépendemment de l'arrivée des mesures. La vérification des mesures envoyées, après échos,

diminue les pertes des mesures en cours de transmission.

Un tampon circulaire est géré pour assouplir le transfert par la liaison série. La saturation du sous-système SS2 et la vitesse du flot de mesure exigent une efficacité dans le transfert et la prise en compte des interruptions générées par l'interféromètre.

Dans le cas où le sous-système SS2 ne répond pas, on peut stocker les mesures sans toutefois dépasser la longueur du tampon. Dans ce cas, on appelle la procédure d'arrêt de l'interféromètre et on sauve la mesure présente sur la liaison parallèle.

Lorsqu'il y a une place, au moins dans le tampon, on appelle la procédure de déblocage de l'interféromètre. Cette gestion de blocage et de déblocage de l'interféromètre rend les pertes de mesures moins probables.

II -3-3 Remarques

La vitesse d'acquisition limite dépend essentiellement de la vitesse de transfert de la liaison série et du dialogue de la gestion de cette liaison. Les deux sous-systèmes SS1 et SS2 ont les mêmes horloges qui gèrent la vitesse de transmission.

La gestion de l'interféromètre peut être modifiée dans l'avenir pour améliorer le transfert des mesures et diminuer les pertes de mesures à très grande vitesse.

II -4 Bruit d'interface matériel du sous-système SS1

L'interface sous système SS1 et l'interféromètre engendre un certain nombre de sources de bruits relevant des composants électroniques utilisées dans la chaîne d'acquisition de mesure.

II -4-1 Architecture globale de l'interface

D'une manière générale, on peut définir cinq niveaux de l'interface :

1 - Le capteur dont la fonction est de convertir en grandeur électrique la grandeur optique à transmettre au sous système SS2.

2 - Un dispositif électronique de sélection permettant de sélecter la grandeur analogique délivrée par le capteur.

3 - Un amplificateur dont le rôle est bien sûr d'amplifier le signal sélectionné et de référencer à la masse mesure le signal différentiel issu du capteur. Selon les performances recherchées, le choix de l'amplificateur sera fait en fonction des caractéristiques exigées. L'architecture de l'interface et sa modularité est en grande partie liée à ce choix de la place de l'amplificateur.

4 - Un convertisseur analogique-digital qui permet de numériser la grandeur analogique de l'interféromètre. Il délivre la grandeur correspondante digitalisée et codée en binaire. Ce convertisseur est parfois précédé d'un bloqueur-échantillonneur qui mémorise sous forme analogique la tension commutée. Il présente donc une grandeur stable à l'entrée du convertisseur pendant le temps de conversion. En outre, il permet un échantillonnage à cadence fixe de la grandeur scrutée en mode de transmission asynchrone avec le sous-système SS2.

5 - Une logique permettant de synchroniser lesopérations de transfert de flots de mesures vers le sous-système SS2.

II -4-2 Impact des modes d'interconnexion des modules sur la précision de l'interface

Les problèmes de raccordement de divers maillons de l'interface ne sont pas négligeables, ils conditionnent énormément la conception de l'architecture générale et sont responsables le plus souvent de dégradation des performances de la précision de l'interface. Il s'agit là d'un problème purement élec-

trique lié à la technologie des câbles, des optoélectroniques, des circuits imprimés et des circuits intégrés et à l'environnement électrique de l'installation.

Plusieurs facteurs physiques influent sur la qualité de transmission de la mesure dans le sous-système SS1. On citera par exemple :

1 - une mauvaise adaptation d'impédance qui est à l'origine d'une perte de niveau du signal ou de l'apparition de bruit supplémentaire. Dans un réseau électrique véhiculant une puissance, les impédances de sources et de charges doivent être adaptées, l'une à l'autre pour assurer un rendement maximal.

2 - les couplages inductifs avec les signaux alternatifs : lorsque le câble transportant la mesure voisine avec un câble de transport de puissance alternative, ce dernier induit dans le cable de mesure une tension proportionnelle à la surface de la boucle formée par le cable transportant la mesure. On utilise pour remédier à ce phénomène comme ligne de transport de mesures des conducteurs torsadés qui réalisent des boucles élémentaires.

3 - les couplages capacitifs : il apparaît un champ électrique de couplage entre deux câbles juxtaposés. Ceci provoque un courant parasite dans le cable de mesure. Afin de pallier cet effet, on isole électrostatiquement les câbles de transport de mesures torsadés par un écrou qui est en général une gaine métallique, tressée et isolée et parfois associé à un troisième conducteur appelé drain, qui réduit la résistance ohmique du blindage.

4 - une disposition incorrecte des prises de masse : il est impératif de disposer d'un point de référence de potentiel fixe, auquel sera le blindage des cables et à partir duquel seront définies les différences de potentiel. On peut concevoir un tel point de référence en réalisant une bonne connexion avec le sol, la terre constituant un condensateur de capacité infinie,

donc à potentiel constant. Les solutions adéquates sont apportées à tous les problèmes de liaison physique des divers maillons de l'interface. Ainsi la qualité du signal est préservée.

II -4-3 Facteurs de qualité de l'interface

Un certain nombre de facteurs définissent la qualité d'un interface d'acquisition de mesures. Nous citons :

1 - la cadence de l'interface

2 - la dynamique des signaux d'entrée

3 - la précision

4 - la sensibilité

5 - la fiabilité

on appelle cadence d'un interface d'acquisition de mesure, sa vitesse d'acquisition des signaux. Elle s'exprime en nombre de voies par seconde ou nombre de points par seconde. La cadence propre du sous-système SS1 est toujours beaucoup plus élevée que la cadence du sous-système SS1 couplé au sous-système SS2, car le sous-système SS2 accomplit deux tâches nécessitant du temps supplémentaire :

- le rangement sur le disque souple du flot de mesures ou sa transmission au sous-système SS3
- le dialogue opérateur-sous-système SS2 dans le cas d'une vérification éventuelle de la mesure.

Ce temps est en fait une mesure de la performance du couplage sous l'aspect de la vitesse et on remarque que dans la pratique il varie dans un rapport de 1 à 1000.

La dynamique d'un signal est la variation maximum du signal d'entrée qui n'altère pas les performances du sous-ensem-

ble sur lequel il a été injecté. Ce paramètre est fonction des gains de tous les sous-ensembles traversés par le signal dans le système.

La précision fixe l'erreur absolue entre grandeurs à mesurer et le résultat de la mesure. Tandis que par contre la sensibilité est mesurée par la variation minimum du signal d'entrée que l'on peut discerner toutes les causes externes de variations demeurant constantes.

La précision reflète à long terme la stabilité de la mesure alors que la sensibilité est liée à la fidélité à court terme.

La fiabilité de l'interface, et en général du système, est sa capacité à remplir plus longtemps sa mission sans panne. L'expression mathématique de la fonction fiabilité est :

$$R(t) = \exp(-\lambda t)$$

c'est la probabilité dans l'intervalle de temps t de n'avoir aucune panne. On définit un temps moyen entre pannes (Mean Time Between Failures ou MTBF) par :

$$MTBF = 1/\lambda$$

La fiabilité décroît de manière exponentielle dans le temps et la moyenne des temps de bon fonctionnement est constante. Le MTBF est donc le temps à l'expiration duquel la probabilité, pour que le système et l'interface considérés fonctionnent encore correctement, est de 0,36.

III - LE SOUS-SYSTEME D'ENTREE-SORTIE SS2 (Fig IV-3)

Ce sous-système est un microordinateur de développement autonome. Il inclut un disque souple pour le stockage des mesures et des programmes de nos applications. Il gère le dialogue

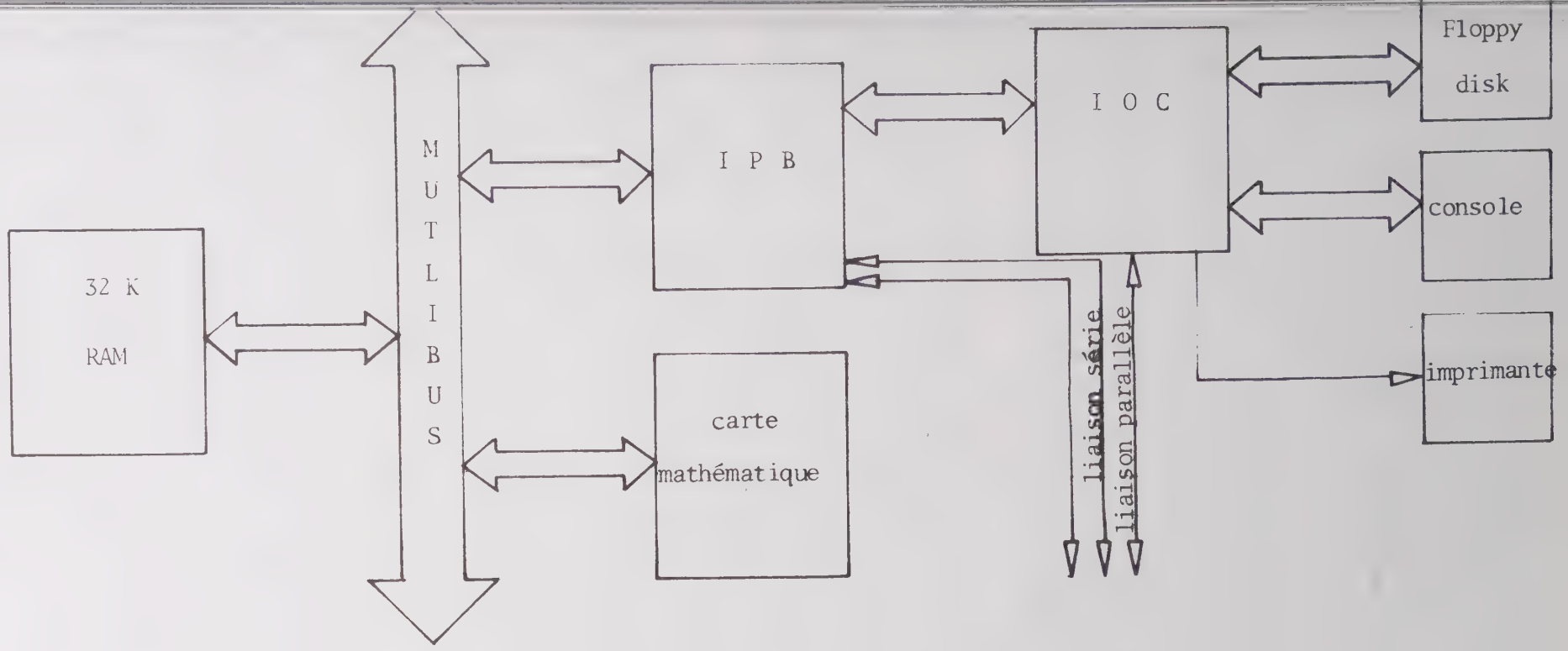

32 K
RAM
M
U
T
L
I
B
U
S
I P B
carte
mathématique
I O C
Floppy
disk
console
imprimante
liaison série
liaison parallèle

FIGURE IV - 3

entre le sous-système SS1 et SS3. Il a la possibilité de calculer une transformation de Fourier rapide sur un nombre de points ne dépassant pas 4 K. Il permet la visualisation des résultats intermédiaires des prétraitements et les résultats finals.

III-1 Les composants matériels

Il est composé de deux cartes microprocesseurs. On prendra la notation suivante :

IPB → la première carte unité centrale maître, appellation anglo-saxonne (Integrated Processeur Board)

IOC → la deuxième carte unité centrale esclave, appellation anglo-saxonne (Input Output Controller)

L'IPB contient son propre microprocesseur : 8080 A-2, de huit bits et de temps de cycle de 2 MHz. Sa RAM est de 32 kilooctets, de temps d'accès de 585 ns. Sa ROM est de 4 kilooctets, de temps d'accès de 450 ns. Il comporte des entrées-sorties séries et des contrôleurs d'interruption. Son architecture est conçue autour d'un bus local.

La deuxième carte ou IOC contient aussi un 8080 A-2 de mêmes caractéristiques que l'IPB. Sa ROM est de huit kilooctets. Il comporte des interfaces d'entrées sorties parallèles.

La communication entre l'IPB et l'IOC est faite par un bus. L'accès du disque souple par l'IPB est faite à travers l'IOC qui utilise un contrôleur intégré.

Le disque souple est d'une capacité de 250 kilooctets et d'une vitesse de transfert de 160 kilobits par seconde. Le temps d'accès piste à piste est de 10 ms maximum.

La fonction de ce sous-système est la gestion du flot de données en circulation dans le système, le stockage des mesures

de l'interféromètre et l'édition des spectres sur un écran ou une imprimante. En outre, le sous-système est doté d'un multibus pour recevoir des cartes esclaves qui peuvent servir à des extension de la mémoire vive. Ces deux extensions nous ont permis d'étendre la RAM de la carte processeur maître à 64 kilooctets et d'avoir un processeur spécialisé de traitement (carte mathématique) des fonctions usuelles. Ces fonctions usuelles sont cablées, elles diminuent le temps de réponse du sous-système d'entrée-sortie dans les calculs en virgule flottante.

L'emploi d'une carte mathématique est lié à la nécessité d'améliorer le temps de réponse de la carte processeur maître dans les pré-traitements de l'interférogramme et du spectre, notamment la transformation de Fourier rapide, l'interpolation et l'édition du spectre.

Le sous-système d'entrée-sortie permet l'amélioration des moyens de test. Ainsi la connaissance de l'interférogramme et du spectre au niveau de chaque sous-système suppose une interaction du sous-système d'entrée-sortie avec l'expérimentateur. Par construction du sous-système, la gestion du dialogue, en utilisant des interruptions, est inexistante. Pour cela, une création de fichier de mesures de l'interférogramme a été choisie pour permettre des tests éventuels de validation de l'expérience.

Disposant d'un matériel à capacité de stockage et de calcul limités, pose des contraintes sur le déroulement et la validation de l'expérience. Contrôler le bon fonctionnement de l'expérience en cours revient à visualiser le spectre de Fourier de l'interférogramme transformé.

En tenant compte de ces données, on a choisi des modes de test divers qui sont implémentés dans le sous-système. Parmi

ces modes, on peut citer :

- acquisition de l'interférogramme
- décodage éventuel des mesures
- transformation de Fourier rapide de 4 Kilomesures
- interpolation du spectre obtenu
- visualisation de chaque résultat trouvé

III-2 Logiciels d'acquisitions de mesures

Pour interfacer le sous-système SS1 et le sous-système SS2, on a introduit un logiciel de gestion dela liaison série et le stockage par des accès du disque souple au moment de l'écriture du fichier de mesure. Cette application est composé de trois parties :

1 - dialogue opérateur —— sous-système SS2
2 - gestion de la liaison série
3 - acquisition et stockage du fichier de mesure

III-3 Logiciels d'applications (Annexe 2ème partie)

Ce logiciel est un pré-traitement des interférogrammes. Il est composé de trois parties dont la troisième est l'édition des résultats finals. On peut les décomposer ainsi :

1 - vérification des mesures stockées dans le disque souple
2 - traitement de la transformation de Fourier de 4 kilomesures
3 - édition du spectre avec interpolation pour une vérification du résultat.

Le choix de ce pré-traitement apporte une aide efficace dans le déroulement de l'enregistrement de l'interférogramme. En effet, le temps pris par l'enregistrement est long et dure environ

une journée. Ainsi continuer à enregistrer sans connaître le résultat préliminaire n'est pas conseillé.

Pour réduire le temps de calcul de la TFR, on a introduit une carte processeur spécialisé ayant les fonctions mathématiques utilisées dans nos calculs.

REMARQUES

1 - Les calculs sont faits en réel, virgule flottante avec un format 32 bits, simple précision.

2 - La mémoire limite nous a imposé un traitement de 4 kilomesures maximum

3 - Le temps de calcul pour ce même nombre de points est de l'ordre d'une minute avec un processeur huit bits et un temps de cycle à peu près égal à 2 µs.

4 - La vitesse de l'édition dépend du nombre de points à imprimer de la vitesse d'impression ou de visualisation sur un écran.

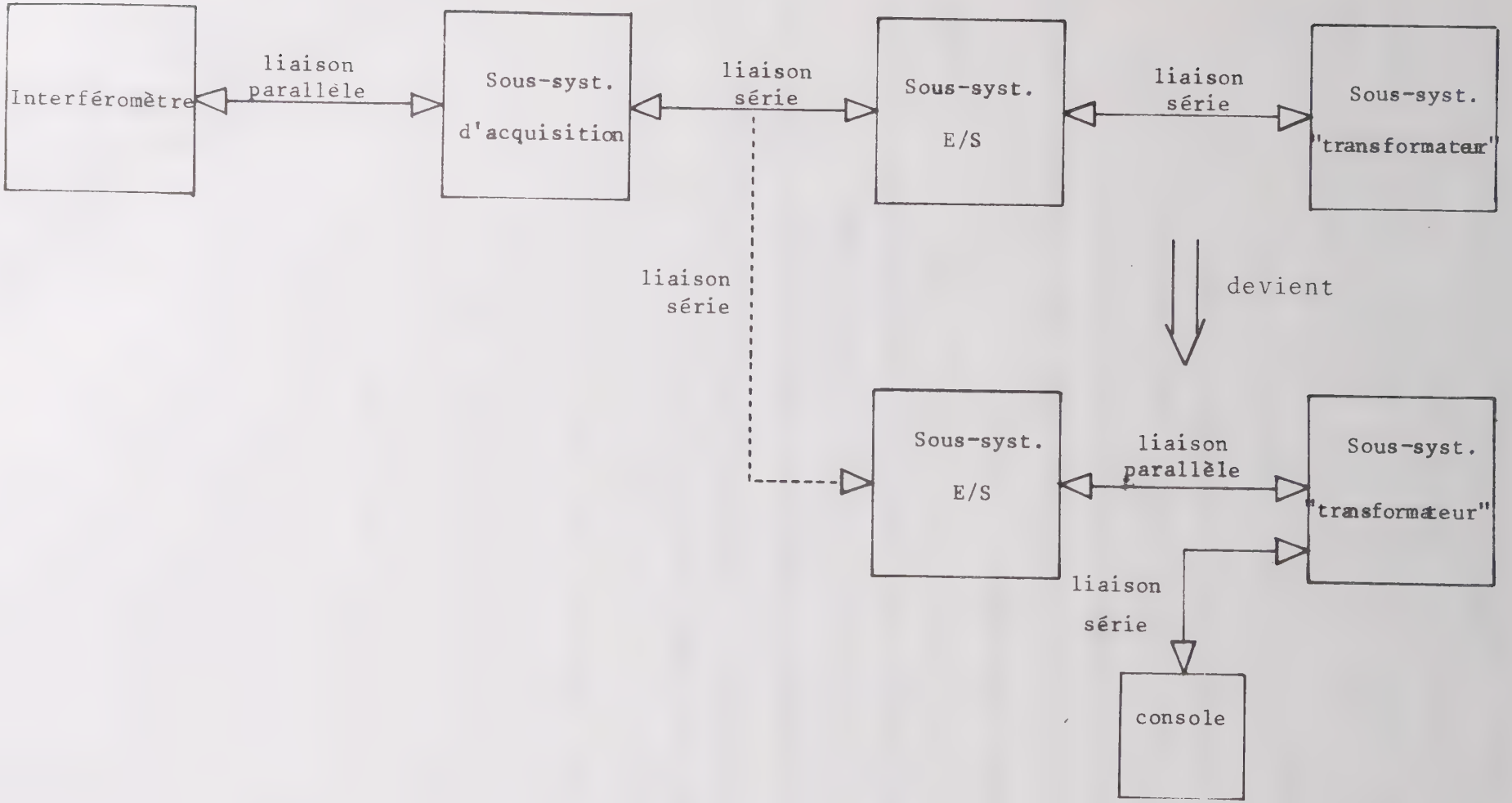

Interféromètre
liaison parallèle
Sous-syst. d'acquisition
liaison série
Sous-syst. E/S
liaison série
Sous-syst. "transformateur"
liaison série
devient
Sous-syst. E/S
liaison parallèle
Sous-syst. "transformateur"
liaison série
console

FIGURE IV - 4

IV - MISE EN OEUVRE DES PROGRAMMES DE L'APPLICATION

IV-1 Logiciel d'acquisition (Annexe 2ème partie)

Il est composé de trois programmes

1- gestion de la ligne parallèle

2- gestion du Tampon circulaire

3- gestion de la ligne série.

La démarche qu'il faut effectuer pour démarrer l'application est la suivante :

Une fois le réglage des conditions de l'expérience et l'électronique fait, on met à zéro le sous système SS1, on charge le programme FFTA dans le système SS2, enfin on fait démarrer l'interféromètre en utilisant l'interrupteur de départ spectromètre.

Les procédures départ et arrêt spectromètre permettent d'interrompre ou d'activer l'interféromètre en fonction de la vitesse d'arrivée des mesures. Le sous système SS1 ne prend pas en compte la gate tant que l'interféromètre n'est pas démarré. Ceci permet à l'expérimentateur d'avancer le chariot pour chercher une différence de marche nulle. Le sous système SS1 commande le départ de l'interféromètre.

Le programme FFTA étant chargé, à l'adresse 3680 H de la mémoire, juste après ISIS, automatiquement l'exécution commence, on voit apparaître à l'écran une suite de "spectro non démarré" tant que le sous système SS2 n'a pas reçu l'information qui indique que l'interféromètre est démarré. Une fois l'interféromètre démarré, l'indication "spectro démarré" apparait sur l'écran. Les deux sous systèmes sont prêts à recevoir des mesures.

A chaque interruption du processeur, la procédure

d'acquisition est activée, la mesure est lue et est stockée dans le tampon circulaire. Si le tampon est plein et que le vidage n'est pas possible, alors on interrompt l'interféromètre en activant la procédure d'interruption adéquate. La transmission entre les deux sous systèmes est en série, et par écho de mesures. Tant que la mesure reçue par le sous système SS2 est bonne, la mesure suivante est envoyée sinon on renvoie la mesure mal reçue.

Le sous système SS2 sauve chaque 512 mesures sur la disquette simple densité. A 4096 mesures, le sous système SS2 est déconnecté du système SS1 pour que l'expérimentateur puisse faire le traitement de l'interférogramme reçu.

IV-2 Le logiciel de traitement (Annexe 2ème partie)

Il est composé de trois programmes :

1- décodage des mesures

2- calcul de la transformée de Fourier adaptée.

3- interpolation.

Le programme codage à l'exécution demande le fichier d'entrée, le fichier de sortie, la gamme comprenant sept valeurs. A la fin de l'exécution, dix premières valeurs sont affichées automatiquement pour vérifier le début de l'interférogramme.

Le programme FFT4KD est exécuté, il demande le fichier décodé d'entrée et le fichier de sortie du spectre. Le fichier de sortie est ouvert en accés direct et l'on écrit par bloc de 1024 mesures. Ceci simplifie la recherche du bloc à interpoler pour des raisons de commodité. On appelle alors INTERP en le chargeant, il demande le fichier à accès direct où le spectre est sauvegardé, la fenêtre d'apodisation pour calculer le transformée de Fourier classique de la fonction d'interpolation. Pour choisir

la fenêtre il suffit de taper un nombre compris entre 0 et 4, la lecture de la fenêtre est alors faite sur un fichier à accès direct où l'on a stocké les cinq fenêtres. Les fenêtres sont calculées une fois pour toute l'application.Le fichier à interpoler contient quatre blocs de 1024 mesures. Le résultat de l'interpolation est constitué par quatre blocs de 5120 mesures.

IV-3 Le logiciel de visualisation (Annexe 2ème partie)

Une fois l'interpolation terminée, on charge le programme EDITE2. A l'exécution il demande le fichier interpolé ou non interpolé à tracer. Si le fichier est interpolé, il demande le numéro du bloc à tracer ainsi que l'unité de sortie. Deux possibilités existent :

- l'imprimante
- l'écran.

On peut tracer l'ensemble des blocs sans recharger le programme.

CONCLUSION

La décomposition du système proposé et réalisé en sous-système constitue un important outil de conception, d'analyse et synthèse à la fois pour la liberté d'abstraction qu'elle permet, et parce qu'elle contribue à la localisation des décisions de conception de mise au point et de test. Nous nous sommes orienté vers l'extensibilité de la topologie du système afin de permettre à l'utilisateur non spécialiste de construire ses algorithmes à quelque niveau de conception ou d'utilisation qu'il se trouve dans le système. Nous avons décrit les moyens d'interfaces entre sous-systèmes et les applications pour le traitement de spectre.

Nous avons conçu deux solutions approchées pour le traitement des spectres :

1 - le traitement limité à 4 K en vue de secourir les décisions de l'utilisateur dans la poursuite de l'expérience.

2 - nous avons donné le moyen d'interpoler et de visualiser les quatre blocs formés du spectre de Fourier.

CONCLUSION

En guise de conclusion : l'évolution de la topologie du système.

Cette première approche du problème posé par le laboratoire de Spectronomie Moléculaire, en fonction des technologies existantes, nous a conduit à réaliser un système ouvert qui, de part sa conception, sa construction et son exploitation est évolutif, extensible et facile à personnaliser.

Nous avons résolu, et ceci dans le cadre du Laboratoire de Spectronomie Moléculaire, le problème principal posé par la vérification du spectre des 4K premiers points de l'interféromètre dans la limite de temps définie au préalable par l'expérimentateur, soit sept minutes.

Le laboratoire dispose ainsi d'une application composée de matériels et de logiciels pour une vérification rapide et sûre de son expérience.

Cette application sera donc constamment accessible et ceci sans remise en cause des investissements antérieurs. L'ultime flexibilité vient de ce que la mise en oeuvre du système est faite à partir d'un système de développement de microprocesseurs et de cartes ISBC. Ainsi, nous n'avons pas bloqué le physicien dans son choix. L'extensibilité des possibilités et des capacités, la possibilité de s'appuyer sur les technologies de l'avenir ; une stratégie planifiée de transposition logicielle protégeant l'investissement d'origine nous permettent de supposer donner une des satisfactions au Laboratoire de Spectronomie Moléculaire.

Les sites informatiques, comme le nôtre, se trouvent confrontés aux problèmes d'augmentation de productivité des micro-ordinateurs : rapidité de calcul pour un grand nombre de mesures.

Pour répondre à ce problème dans l'avenir, il faut ajou-

ter un autre sous-système connecté directement au sous-système d'acquisition, employant un microprocesseur 16 bits, une forte extension mémoire, un disque dur etc..... Par exemple, la solution consistant à utiliser une carte SBC 86/12, un coprocesseur mathématique 8087 et une unité de mémoire de 512 K octets nous paraît très abordable à court terme. En effet, pour traiter au-delà de 128 K mesures, le Laboratoire fait appel au calculateur du CIRCE. Dans le cas contraire, cette extension est très intéressante et son coût du point de vue matériel et logiciel est accessible.

La carte SBC 86/12 est organisée autour d'un bus qui inclut vingt bits d'adresse et seize bits de données. La carte comprend un processeur 8086, avec une horloge de cinq mégahertz. En outre, il comporte une RAM de trente-deux kilooctets à double accès, une ROM de seize kilooctets. La capacité d'adressage du processeur 8086 est de un million d'octets. Une liaison série et une autre parallèle différente du multibus permettent l'interfaçage avec les deux sous-systèmes SS1 et SS2 (Figures IV-5, IV-6). Le multibus lui permet d'accéder aux extensions mémoires prévues pour les traitements à grande échelle.

Deux cas sont à considérer pour l'accès de la mémoire vive :

- accès par le CPU interne de la carte processeur
- accès par le multibus

Ainsi, on peut accéder à des parties de la RAM par l'interface Multibus. Les sémaphores matériels existant au sein de la carte permettent la protection d'une partie ou de la totalité de la RAM.

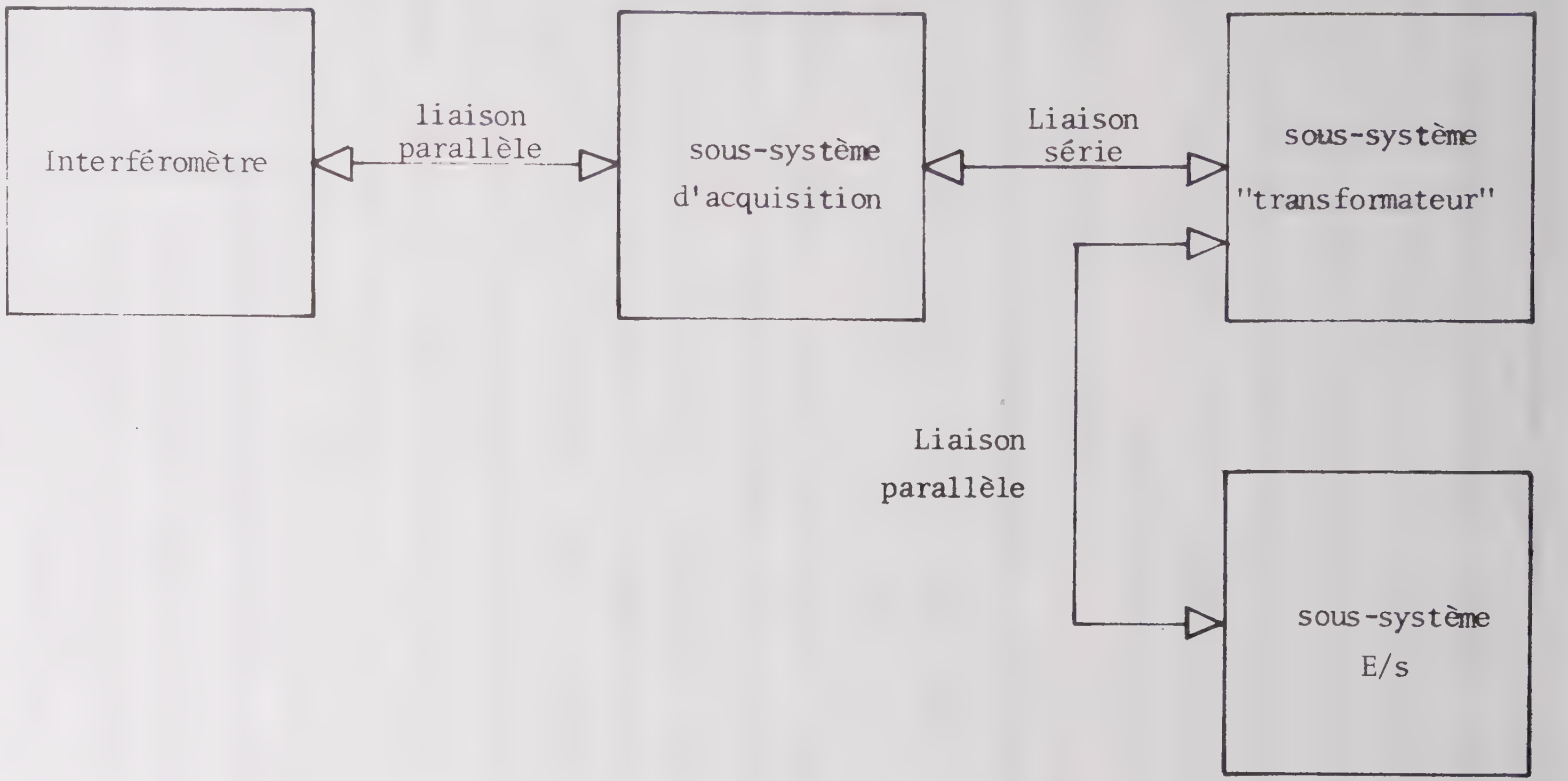

Interféromètre
liaison
parallèle
sous-système
d'acquisition
Liaison
série
sous-système
"transformateur"
Liaison
parallèle
sous-système
E/s

FIGURE C - 1

Le multi-traitement ou le traitement parallèle est une caractéristique importante de la carte SBC 86/12. Par ailleurs toute la logique nécessaire au fonctionnement de la carte dans un environnement multimaître y est incluse. Elle peut donc travailler en parallèle avec d'autres modules. Le multibus n'est demandé par un module maître que lorsqu'une ressource commune du système est nécessaire.

La structure du multibus, qui est un bus standard proposé par Intel, est constituée des fonctions suivantes :

- l'interconnexion modulaire et la communication entre les différentes cartes de la famille SBC.
- la connexion de plusieurs modules processeurs travaillant en parallèle. Le multibus permet d'augmenter les performances et la puissance de traitement en minimisant les conflits sur le bus
- le partage des ressources. Plusieurs cartes processeurs SBC peuvent partager des ressources communes sur le multibus.

Pour mieux comprendre l'intérêt du multibus, un parallèle avec les bus traditionnels est nécessaire. Un système qui utilise un bus traditionnel est en conflit permanent quant il s'agit d'utiliser les ressources mémoires où d'entrées-sorties. L'addition d'une unité centrale ne représente pas une augmentation de la performance mais une dégradation qui est due aux conflits trop importants sur le bus, ceci implique une perte de temps sur le bus. Avec le multibus, l'augmentation de performances est rendue possible car chaque carte SBC possède ses propres ressources. Une unité centrale demandera le contrôle du multibus seulement dans le cas où la ressource désirée est extérieure, ce qui réduit considérablement les demandes de service du multibus. Toutes les ressources résidant d'une carte SBC sont protégés par des sémaphores matériels contre les accès provenant du multibus.

Dans le cas où plusieurs cartes SBC sont connectées

sur le multibus, il faut établir une hiérarchie entre ces différentes cartes. Une carte maître peut, en fonction de sa priorité sur le multibus, contrôler celui-ci. Une carte esclave ne peut en aucun cas contrôler le multibus. Cette carte est une ressource commune aux différentes cartes maîtres.

La gestion des priorités sur le multibus est intégrée sur chaque carte maître. Elle permet d'allouer une ressource du multibus à une carte maître si celui-ci a la plus forte priorité.

La gestion des priorités sur le multibus peut être résolue en série ou en parallèle.

Jusqu'à présent, seul le côté matériel a été examiné. Il est évident qu'autour d'un système multiprocesseurs utilisant le multibus, des considérations logicielles sont à prendre en compte.

Bien que ce paragraphe ne soit pas axé sur les considérations trop avancées du logiciel de gestion du multibus, quelques mots sont cependant nécessaires afin de dégager certaines contraintes à résoudre dans un proche avenir.

Il existe trois composantes essentielles :

- l'exclusion mutuelle
- la synchronisation
- la communication entre différents processeurs maîtres

Plusieurs ressources communes, telles mémoires sont partagées par les différents processeurs. L'exclusion mutuelle est la possibilité logicielle qu'a chacun des processeurs de pouvoir verrouiller le multibus pendant une section critique du programme.

Cette conception permet non seulement uneprotection de l'investissement, un accroissement important des performances et la configuration optimale pour le projet du Laboratoire avec la possibilité de la faire évoluer. De plus, la liaison directe entre les deux cartes SBC assurera une meilleure répartition des taches du système de développement.

La transposition du logiciel de notre application du sous-système SS2 au sous-système SS3 nous a permis d'augmenter deux paramètres essentiels :

- la vitesse de traitement
- la capacité

L'appellation "transformateur" est due à la fonction principale qu'exécute le sous-système SS3. Cette fonction est la transformation de Fourier rapide étudiée au chapitre I.

V-4 Interface SS2 et SS3 (Figure C-1)

Pour permettre un développement de nos applications sur la carte SBC, on a introduit un logiciel utilitaire pour gérer l'interface. Deux possibilités de modes d'interfaçage sont utilisées :

- la liaison parallèle
- la liaison série

L'application ainsi réalisée autorise une continuelle augmentation des performances du matériel existant et permet d'offrir un support de développement répondant aux familles de microprocesseurs actuelles et à venir.

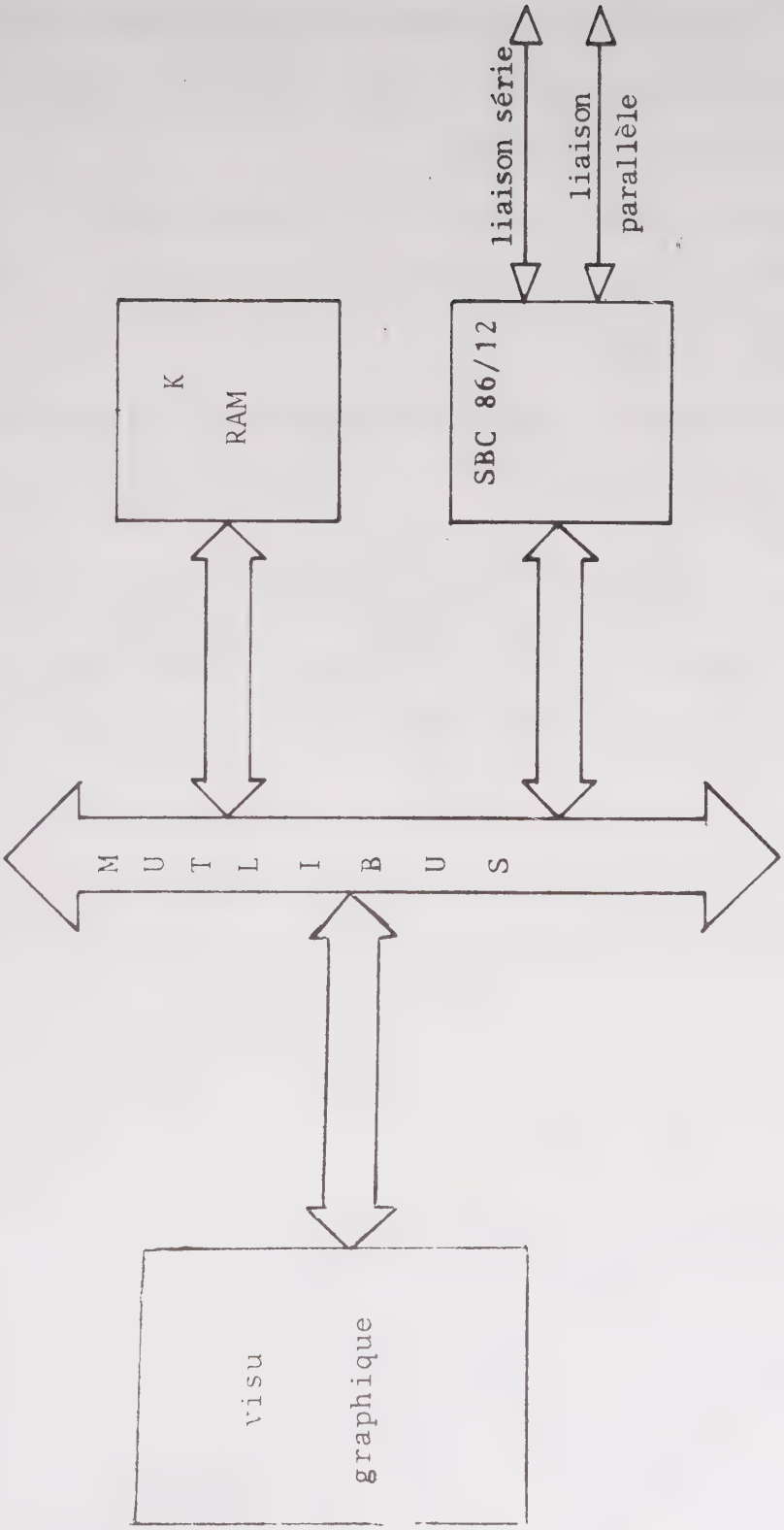

liaison série
liaison parallèle
K RAM
SBC 86/12
MULTIBUS
visu graphique

FIGURE C - 2

Nous venons de voir que la motivation principale du système est la facilité d'accéder à ses parties aisément. L'évolution de la topologie du système intervient au niveau du sous-système SS3. Cette évolution est d'autant plus justifiée que la transformation de Fourier rapide deviendrait une réalité.

Ainsi ce système augmentera les performances et la capacité de traitement du sous-système SS3. L'évolution sera dirigiée sur deux axes principaux :

1 - un environnement matériel bien conçu autour de la carte SBC 86/12. Ceci nous permettra d'atteindre la modularité optimale (Figure C - 2).

2 - un environnement logiciel modulaire, cette possibilité nous fera parvenir à des systèmes faciles à mettre en oeuvre.

Ces deux chemins nous permettront de mettre en oeuvre les traitements rapides de nos algorithmes. Pour cela, une étude doit être entreprise afin d'approfondir l'évolution de l'application.

ANNEXE

1ÈRE PARTIE

2ÈME PARTIE

3ÈME PARTIE

1ère PARTIE

I- ALGORITHMES

Pour comprendre la différence entre une transformation de Fourier rapide séquentielle et une transformation de Fourier rapide parallèle, on est amené à rappeler la méthode de calcul de TFR en séquentiel.

II-1 Définition

On peut noter la TFR par la relation suivante

$$Y(n) = \sum_{k=0}^{N-1} x_o(k)\ W^{nk} \qquad (23)$$

$n=0,1,\ldots,N-1$

$k=0,1,\ldots,N-1$

$W = e^{-2j\pi/N}$

la fonction $x_o(k)$ ε au corps des complexes

la fonction $Y(n)$ ε au corps des complexes

I-2 Algorithme de la TFR

L'algorithme de la TFR a été explicité dans le chapitre I en utilisant des formules mathématiques calculant les parties réelles et parties imaginaires de Y(n).

On a traité le cas de $N=2^{\gamma}$ (N étant le nombre de points à transformer). Pour une base quelconque, on se réfère à la littérature abondante dans ce domaine, en s'appuyant sur les références données dans la bibliographie.

Il existe deux niveaux de traitement pour une fonction appartenant au corps des complexes et quatre niveaux pour une fonction appartenant au corps des réels.

I-2-1 Fonction appartenant au corps des complexes

De la définition

$$Y(n) = \sum_{k=0}^{N-1} x_o(k)\ W^{nk} \qquad n\varepsilon[0,N-1]$$

on déduit, en utilisant la base 2 et en posant $N=2^{\gamma}$, que Y(n) est calculé au bout de γ étapes de calcul. Mais on n'obtient pas l'organisation des données souhaitée. L'ordre dans lequel doivent être rangées les valeurs Y(n) après les étapes de calcul est donné par la régle dite "d'inversion de bits" [1].

Les deux niveaux sont constitués alors par :

-1. γ étapes de calcul.

-2. 1 étape d'ordonnancement.

I-2-2 Fonction appartenant au corps des réels

De
$$Y(n) = \sum_{k=0}^{N-1} x_o(k)\ e^{-j2\pi nk/N} \qquad (24)$$

$$= \sum_{k=0}^{N-1} x_o(k)\ W^{nk} \qquad W=e^{-j2\pi/N}$$

$$n\varepsilon[0,N-1]$$

et
$$x_o(k) = a_o(k) + jb_o(k)$$

on a
$$x_o(k) = a_o(k) \qquad b_o(k) = 0$$

et
$$Y(n) = \sum_{k=0}^{N-1} a_o(k)\ e^{-j2\pi nk/N} \qquad (25)$$

$a_o(k)$ et $b_o(k)$ étant des fonctions appartenant au corps des réels.

On utilise l'algorithme de calcul d'une fonction complexe pour calculer une fonction réelle tel que la partie imaginaire est remplie par des valeurs de la fonction de départ décalées d'un rang.

I-2-3 Formule des recombinaisons dans le cas d'une fonction réelle

Pour transformer 2N points en utilisant l'algorithme N points, il suffit de remarquer que

$$\begin{aligned} h(k) &= x(2k) \\ g(k) &= x(2k+1) \end{aligned} \qquad k = 0,1,\ldots,N-1 \qquad (26)$$

en effet

$$Y(n) = \sum_{k=0}^{2N-1} x(k)\, e^{-j2\pi nk/2N}$$

$$= \sum_{k=0}^{N-1} x(2k)\, e^{-j2\pi nk/2N} + \sum_{k=0}^{N-1} x(2k+1) e^{-j2\pi n(2k+1)/2N}$$

$$= H(n) + e^{-j\pi n/N}\, G(n)$$

d'où

$$Y(n) = \sum_{k=0}^{N-1} y(k)\, e^{-j2\pi nk/N} \qquad \text{avec } y(k)=h(k) + jg(k)$$

$$= R(n) + jI(n)$$

Il est intéressant de remarquer qu'une fonction réelle ou complexe peut se décomposer en deux fonctions, une paire, l'autre impaire, d'où :

$$f(k) = \frac{f(k)}{2} + \frac{f(k)}{2} = \left[\frac{f(k)}{2} + \frac{f(-k)}{2}\right] + \left[\frac{f(k)}{2} - \frac{f(-k)}{2}\right] \qquad (27)$$

f(k) étant périodique, de période N, alors

d'où

$$f_e(k) = \frac{f(k)}{2} + \frac{f(N-k)}{2} \qquad \text{ou paire}$$
$$f_o(k) = \frac{f(k)}{2} - \frac{f(N-k)}{2} \qquad \text{ou impaire} \tag{28}$$

si F(n) est la transformée discrète de f(k), on peut écrire

$$F(n) = R(n) + jI(n) = F_e(n) + F_o(n) \tag{29}$$

où $F_e(n) = R(n)$ $\quad F_o(n) = jI(n)$

Pour transformer deux fonctions réelles en utilisant une fonction complexe, on écrit

$$f(k) = h(k) + jg(k) \tag{30}$$

la transformation de Fourier rapide étant linéaire, alors

$$\begin{aligned} F(n) &= H(n) + jG(n) \\ &= \left[H_r(n) + jH_i(n)\right] + j\left[G_r(n) + jG_i(n)\right] \\ &= R(n) + jI(n) \\ &= \left(\frac{R(n)}{2} + \frac{R(N-n)}{2}\right) + \left(\frac{R(n)}{2} - \frac{R(N-n)}{2}\right) \\ &\quad + j\left(\frac{I(n)}{2} + \frac{I(N-n)}{2}\right) + j\left(\frac{I(n)}{2} - \frac{I(N-n)}{2}\right) \end{aligned} \tag{31}$$

En utilisant ces relations, on peut alors écrire :

$$H(n) = \left[\frac{R(n)}{2} + \frac{R(N-n)}{2}\right] + j\left[\frac{I(n)}{2} - \frac{I(N-n)}{2}\right]$$
$$G(n) = \left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right] - j\left[\frac{R(n)}{2} - \frac{R(N-n)}{2}\right] \tag{32}$$

Des formules (32), on peut calculer la partie réelle et la partie imaginaire en utilisant la formule suivante :

$$Y(n) = H(n) + e^{-j\pi n/N} \; G(n)$$

on a alors :

$$XR_1(n) = \frac{R(n)}{2} + \frac{R(N-n)}{2} + \cos\frac{\pi n}{N}\left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right] \qquad (47a)$$

$$- \sin\frac{\pi n}{N}\left[\frac{R(n)}{2} - \frac{R(N-n)}{2}\right] \qquad n=0,1,\ldots,N-1$$

$$XI_1(n) = \frac{I(n)}{2} - \frac{I(N-n)}{2} - \sin\frac{\pi n}{N}\left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right] \qquad (47b)$$

$$- \cos\frac{\pi n}{N}\left[\frac{R(n)}{2} - \frac{R(N-n)}{2}\right] \qquad n=0,1,\ldots,N-1$$

II - ALGORITHMES DE TFR, CONVOLUTION ET CORRELATION

II -1 Algorithme de la transformation de Fourier rapide pour une fonction complexe.

Des équations suivantes :

$$x_\ell(k) = x_{\ell-1}(k) + W_N^{nk}\, x_{\ell-1}(k + \frac{N}{2^\ell})$$

$$x_\ell(k + \frac{N}{2^\ell}) = x_{\ell-1}(k) - W_N^{nk}\, x_{\ell-1}(k + \frac{N}{2^\ell})$$

$\ell = 1,m$ indice d'étape

$k = 0,N-1$

$n = 0,N-1$

$$Y(n) = \sum_{k=0}^{N-1} X(k)\, W_N^{nk} \qquad \text{où} \quad W = e^{-j\frac{2\pi}{N}}$$

on a :

$$a_\ell(k) = a_{\ell-1}(k) + c(k)(k + \frac{N}{2^\ell}) - S(k)b_{\ell-1}(k + \frac{N}{2^\ell})$$

$$b_\ell(k) = b_{\ell-1}(k) + c(k)b_{\ell-1}(k + \frac{N}{2^\ell}) + S(k)a_{\ell-1}(k + \frac{N}{2^\ell})$$

1. Formation de $x_o(k)$ $\quad k = 0,1,\ldots,N-1$

2. Calcul de $x_\ell(k)$ $\quad \ell = 1,m \quad ; \; k = 0,1,\ldots,N-1$

   Faire sur ℓ et k

   $$x_\ell(k) = x_{\ell-1}(k) + W_N^{nk}\, x_{\ell-1}(k + \frac{N}{2^\ell})$$

   $$x_\ell(k + \frac{N}{2^\ell}) = x_{\ell-1}(k) - W_N^{nk}\, x_{\ell-1}(k + \frac{N}{2^\ell})$$

   Fin faire sur ℓ et k

   On peut s'inspirer du calcul de $a_\ell(k)$ et $b_\ell(k)$

II -2 Calcul de la TFR de deux fonctions réelles simultanément

1. Lire les fonctions g(k), h(k) k=0,1,...,N-1

2. Formuler la relation suivante

   $y(k) = g(k) + jh(k) \qquad k = 0,1,\ldots,N-1$

3. Calcul de

   $$Y(n) = \sum_{k=0}^{N-1} y(k)\, e^{-j2\pi\frac{nk}{N}}$$

   $$= R(n) + jI(n) \qquad n=0,1,\ldots,N-1$$

   où R(n) et I(n) sont respectivement la partie réelle et la partie imaginaire de Y(n).

4. Calcul de

   $$H(n) = \left[\frac{R(n)}{2} + \frac{R(N-n)}{2}\right] + j\left[\frac{I(n)}{2} - \frac{I(N-n)}{2}\right]$$

   $$G(n) = \left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right] - j\left[\frac{R(n)}{2} - \frac{R(N-n)}{2}\right]$$

   où H(n) et G(n) sont les transformées de Fourier discrètes de h(k) et g(k) respectivement.

II -3 Transformation de 2N échantillons utilisant une de N échantillons

1. Lire $x(k)$ $\quad k = 0,1,\ldots, 2N-1$

2. Formuler $x(k)$ en :

$$h(k) = x(2k) \qquad k = 0,1,\ldots,N-1$$

$$g(k) = x(2k+1) \qquad k = 0,1,\ldots,N-1$$

3. Former la fonction complexe tel que :

$$y(k) = h(k) + jg(k) \qquad k=0,1,\ldots,N-1$$

4. Calcul de

$$Y(n) = \sum_{k=0}^{N-1} y(k)\, e^{-j2\pi\frac{nk}{N}}$$

$$= R(n) + jI(n) \quad n = 0,1,\ldots,N-1$$

où $R(n)$ et $I(n)$ sont respectivement la partie réelle et la partie imaginaire de $Y(n)$.

5. Calcul de

$$X_r(n) = \left[\frac{R(n)}{2} + \frac{R(N-n)}{2}\right] + \cos\frac{\pi n}{N}\left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right]$$

$$- \sin\frac{\pi n}{N}\left[\frac{R(n)}{2} - \frac{R(N-n)}{2}\right]$$

$$n = 0,1,\ldots,N-1$$

$$X_i(n) = \left[\frac{I(n)}{2} - \frac{I(N-n)}{2}\right] - \sin\frac{\pi n}{N}\left[\frac{I(n)}{2} + \frac{I(N-n)}{2}\right]$$

$$- \cos\frac{\pi n}{N}\left[\frac{R(n)}{2} - \frac{R(N-n)}{2}\right]$$

$$n = 0,1,\ldots,N$$

où $X_r(n)$ et $X_i(n)$ sont respectivement la partie réelle et la partie imaginaire de la transformée de $x(k)$

II -4 Algorithme de convolution en utilisant la FFT

1. Lire $f_1(k)$, $f_2(k)$
2. Décaler $f_1(k)$ et $f_2(k)$ de l'origine d'une quantité a et b respectivement.
3. Formuler $f_1(k)$ et $f_2(k)$ par

   $f_1(k) = f_1(kT + a)$ $\quad k = 0,1,\ldots,P-1$

   $f_2(k) = f_2(kT + b)$ $\quad k = 0,1,\ldots,Q-1$

   T période de l'échantillonnage de $f_1(t)$ et $f_2(t)$
4. Choisir N de telle sorte que les relations suivantes

   $N > P + Q - 1$

   $N = 2^{\gamma}$ $\quad \gamma$ entier

   soient satisfaites

   où P est le nombre d'échantillons définissant $f_1(t)$ et Q celui de $f_2(t)$
5. Remplir avec des zéros les fonctions échantillonnées de l'étape (3)

   $f_1(k) = 0$ $\quad k = P, P+1,\ldots,N-1$

   $f_2(k) = 0$ $\quad k = Q, Q+1,\ldots,N-1$
6. Calcul de la transformation discrète de $f_1(k)$ et $f_2(k)$

$$X(n) = \sum_{k=0}^{N-1} f_1(k)\ e^{-j2\pi\frac{nk}{N}}$$

$$H(n) = \sum_{k=0}^{N-1} f_2(k)\ e^{-j2\pi\frac{nk}{N}}$$

7. Calcul du produit normal

$$Y(n) = X(n)\ H(n)$$

8. Calcul de la transformée inverse

$$y(k) = \sum_{n=0}^{N-1} \left[\frac{1}{N}\ Y^{*}(n)\right]\ e^{-j2\pi\frac{nk}{N}}$$

II -5 Algorithme de corrélation en utilisant la FFT

1. Lire $f_1(k)$, $f_2(k)$
2. Décaler $f_1(k)$ et $f_2(k)$ de l'origine d'une quantité a et b respectivement
3. Choisir N pour satisfaire les relations suivantes

   $N > P + Q - 1$

   $N = 2^{\gamma}$ γ entier
4. Définir $f_1(k)$ et $f_2(k)$ telles que

   $f_1(k) = 0 \qquad k = 0,1,\ldots, N-P$

   $f_1(k) = f_1(kT+a) \qquad k = N-P+1, N-P+2, \ldots, N-1$

   $f_2(k) = f_2(kT+b) \qquad k = 0,1,\ldots,Q-1$

   $f_2(k) = 0 \qquad k = Q,Q+1,\ldots,N-1$
5. Calculer la transformée discrète de $f_1(k)$ et $f_2(k)$

$$X(n) = \sum_{k=0}^{N-1} f_1(k)\, e^{-j2\pi\frac{nk}{N}}$$

$$H(n) = \sum_{k=0}^{N-1} f_2(k)\, e^{-j2\pi\frac{nk}{N}}$$

6. Changer le signe de la partie imaginaire de H(n) pour obtenir $H^*(n)$
7. Calculer le produit

$$Z(n) = X(n)\, H^*(n)$$

8. Calculer la transformée inverse

$$z(k) = \sum_{k=0}^{N-1} \left(\frac{1}{N} Z^*(n)\right) e^{-j2\pi\frac{nk}{N}}$$

2ème PARTIE

MACHINE LAHYANE

PROGRAMME ACQUISITION
ET TRAITEMENT D'INTERFEROGRAMME

Objet : Sous programme du traitement local d'interférogramme et du spectre qui en résulte.

Nombre de pages : 36

Date d'édition : 1er septembre 1982

Référence : Lahyane Version 1.0

A - Sous programmes d'acquisition d'une mesure

B - Sous programmes de gestion de la transmission de données

C - Sous programmes de mise en forme de données

D - Sous programmes de transformation de Fourier rapide

E - Sous programmes de traitement du spectre

F - Sous programmes d'édition.

A - Sous-programme d'acquisition d'une mesure

1- INIT : initialisation de la liaison série et liaison parallèle

2- ARRET$SPECTRO : envoi de l'ordre d'arrêt de l'interféromètre

3- DEPART$SPECTRO : envoie l'ordre de départ de l'interféromètre à travers la liaison parallèle

4- ACQUISITION : réception des quatre octets de la mesure de la liaison parallèle

5- INTERRUPTION : prise en compte de la gate et présence d'une mesure à travers la liaison parallèle

6- FIN$TRANSMISSION : fin de transmission des mesures

7- SPECTRO$PRET : détection de la présence de l'interféromètre à travers la liaison parallèle.

8- PUT,GET : gèrent le buffer circulaire

B - Sous-programme de gestion de la transmission des données

1- EMISSION : envoi et réception de la mesure côté SBC80/10

2- ECHO : vérification de la mesure envoyée

3- SEND : envoi d'un caractère côté SBC80/10

4- RECEIVE : réception d'un caractère côté SBC80/10

5- SEND : envoi d'un caractère côté MDS 221

6- RECEIVE : réception d'un caractère côté MDS 221

7- V9600 : initialise la ligne série à 9600 band

8- TTYIN : réception d'un caractère

9- FIN : fin de transmission

10- ECRITURE$DISQUE : écrit un buffer sur la disquette.

C - Sous programme de mise en forme de données

1- SAVFOR,READPL : lit fichier système, écrit fichier Fortran

2- DECSAV,DECODE : décode la mesure.

D - Sous programme de transformation de Fourier rapide

1- FFTON : programme principal

2- FFT4KM : transformation de Fourier rapide d'une fonction réelle impaire

3- NORM : normalisation du spectre à la valeur 1

4- HERM1 : calcul de la fonction hermitique

5- HERM2 :

6- TRIAGE :

7- COOLEY : algorithme de Cooley-Tukey

8- COOLE 1 :

9- COOLE 4 calcul d'une moitié du spectre de Fourier

10- ANGLE :

E - Sous programme de traitement du spectre

1- INTER1 : programme principal

2- FILTRE : transformation de Fourier de la fonction d'apodisation

3- INTER3 : interpolation d'un bloc de 1024 mesures

4- FENETO : interpolation sans apodisation et transformation de Fourier

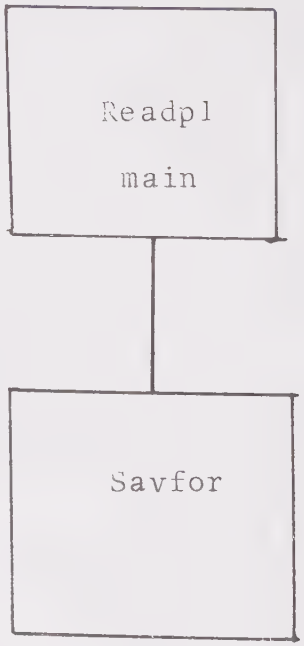
5- FENETR : choix de l'apodisation

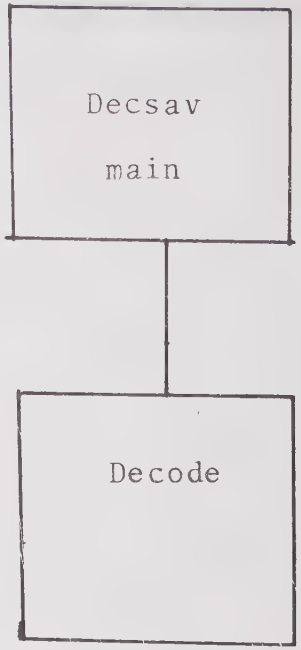
F - Sous programme d'édition

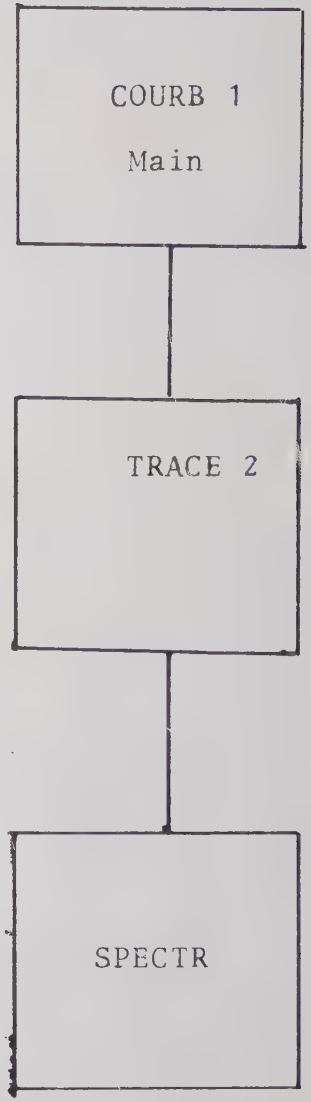
1- COURB1 : programme principal

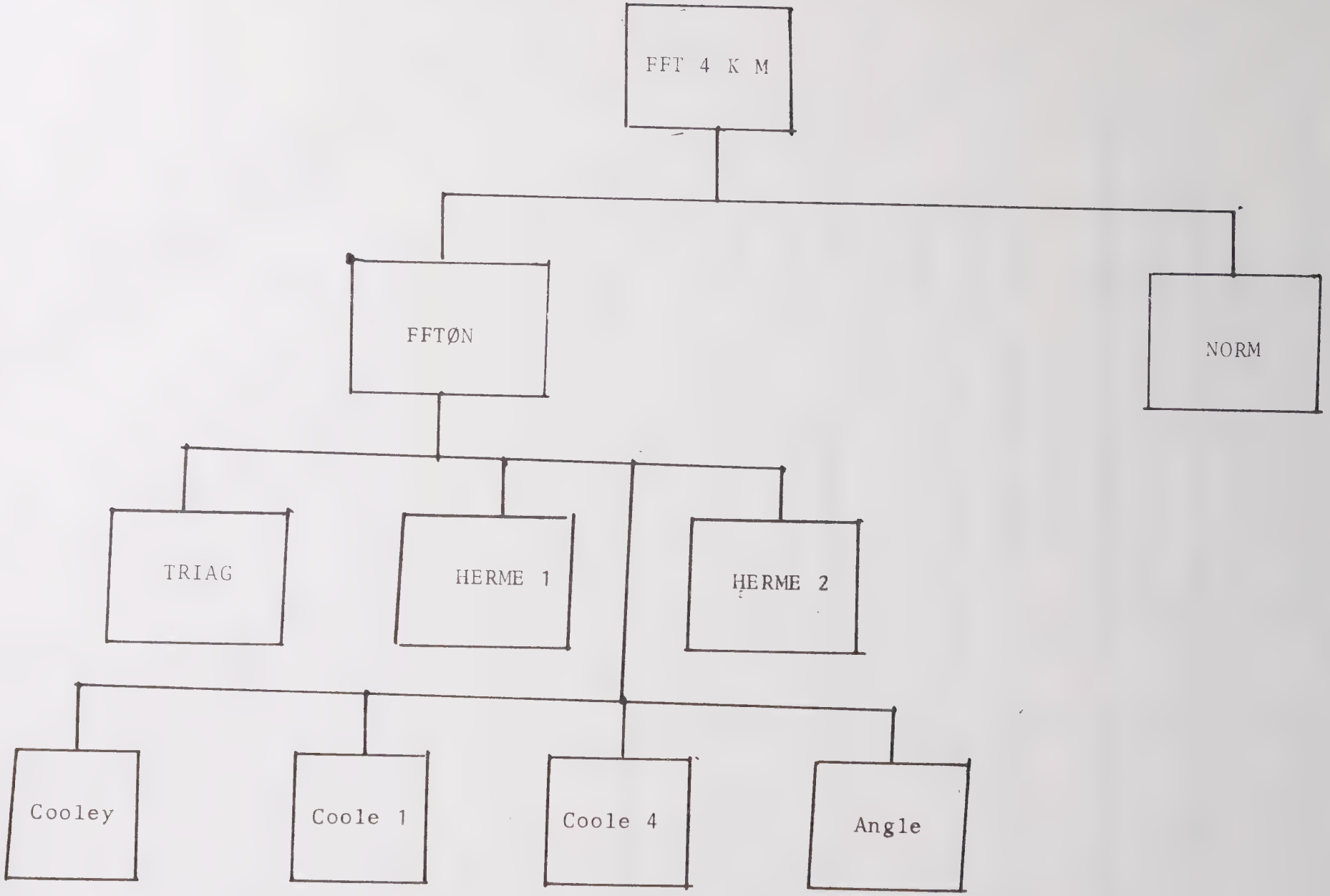
2- TRACE2 : choix d'une partie du spectre

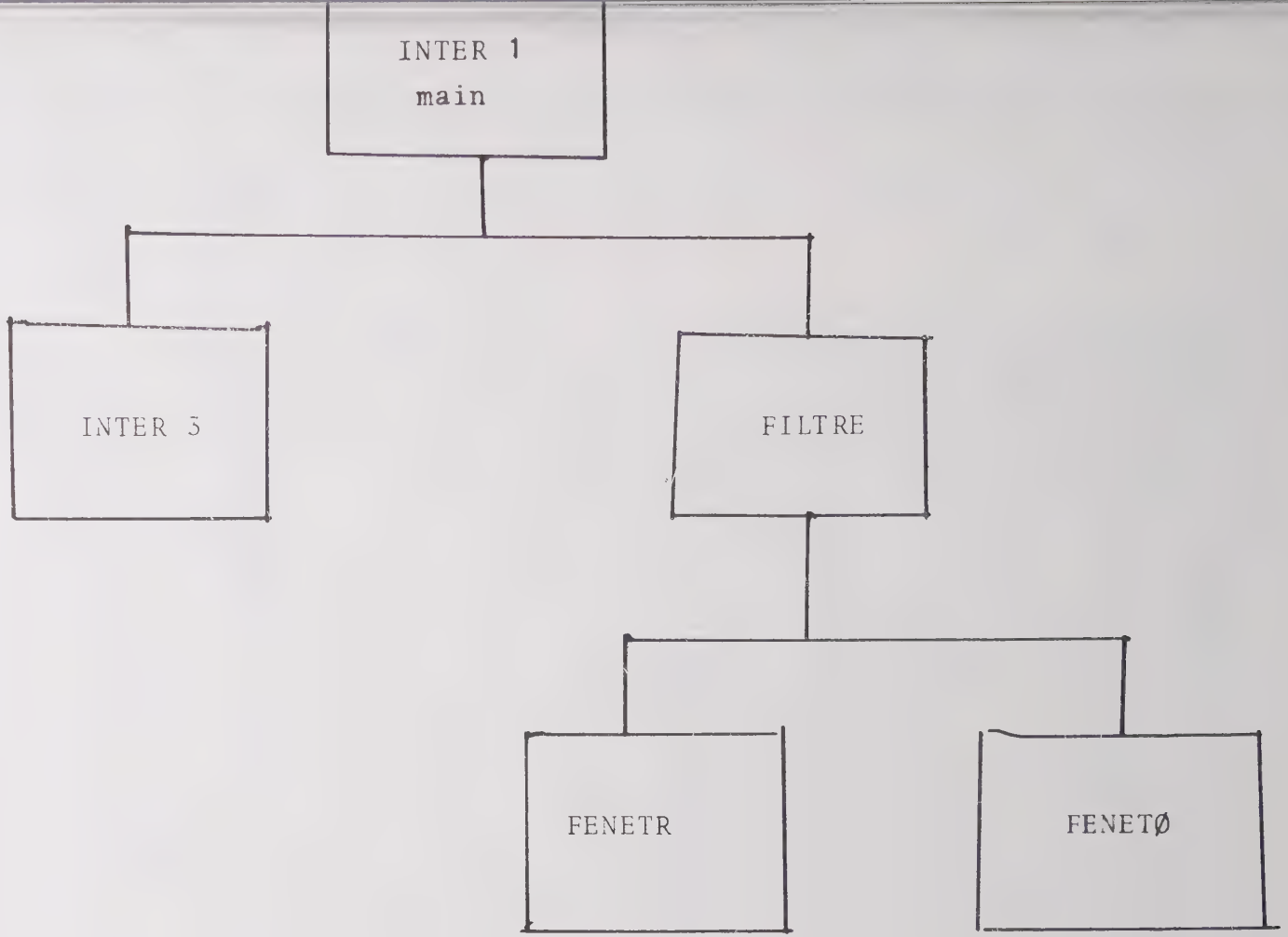
3- SPECTR : interpolé à tracer


Readpl
main
Savfor


Decsav
main
Decode


COURB 1
Main
TRACE 2
SPECTR


FFT 4 K M
FFTØN
NORM
TRIAG
HERME 1
HERME 2
Cooley
Coole 1
Coole 4
Angle


INTER 1
main
INTER 3
FILTRE
FENETR
FENETØ

```
  2      $FREEFORM
  3         SUBROUTINE SAVFOR
  4           CHARACTER FICHS*10,IMP
  5           DIMENSION NTAB(4096)
  6           COMMON NTAB
  7           WRITE(6,66)
  8      66    FORMAT('FS=',$)
  9           READ(5,'(A10)')FICHS
 10           WRITE(6,67)
 11      67    FORMAT('VERIFY?',$)
 12           READ(5,'(A1)')IMP
 13            IF (IMP.EQ.'Y') THEN
 14                WRITE(6,'(12(Z4,2X))')NTAB
 15            ELSE
 16            END IF
 17            OPEN(9,FILE=FICHS,FORM='UNFORMATTED',
     &              ACCESS='SEQUENTIAL',STATUS='NEW')
 18            WRITE(9)NTAB
 19            CLOSE(9)
 20            IF (IMP.EQ.'Y') THEN
 21               OPEN(7,FILE=FICHS,FORM='UNFORMATTED',
     &                 ACCESS='SEQUENTIAL',STATUS='OLD')
 22               READ(7)NTAB
 23               CLOSE(7)
 24               WRITE(6,'(12(Z4,2X))')NTAB
 25            ELSE
 26            END IF
 27          RETURN
 28         END


MODULE INFORMATION:

     CODE AREA SIZE      = 0273H     627D
     VARIABLE AREA SIZE  = 002FH      47D
     MAXIMUM STACK SIZE  = 000AH      10D
     30 LINES READ

     0 PROGRAM ERROR(S) IN PROGRAM UNIT SAVFOR

     0 TOTAL PROGRAM ERROR(S)
     END OF FORTRAN COMPILATION
```

```
      $FREEFORM
         CHARACTER FICHE*10
         CHARACTER FICHS*10,IMP
          DIMENSION NT(4096),T(4096),IGAM(7)
           EQUIVALENCE (NT(1),T(2049))
           COMMON IGAM
           WRITE(6,60)
60          FORMAT('FE=',$)
           READ(5,'(A10)')FICHE
           WRITE(6,65)
65          FORMAT('FS=',$)
          READ(5,'(A10)')FICHS
          OPEN(9,FILE=FICHE,ACCESS='SEQUENTIAL',
     &          FORM='UNFORMATTED',STATUS='OLD')
          OPEN(12,FILE=FICHS,ACCESS='SEQUENTIAL',
     &          FORM='UNFORMATTED',STATUS='NEW')
          WRITE(6,62)
62         FORMAT('GAMME?')
           DO 1 I=1,7
1           READ(5,'(I5)')IGAM(I)
            READ(9)NT
            DO 2 I=1,4096
2             CALL DECODE(NT(I),T(I))
            WRITE(12)T
           CLOSE(9)
           CLOSE(12)
          STOP
         END


      $FREEFORM
        SUBROUTINE DECODE(NN,R)
         DIMENSION IGAM(7)
          INTEGER*1 A(4)
          EQUIVALENCE (A(1),NT2),(A(3),NT1)
          COMMON /GAMME/IGAM
           NT2=NN
           A(3)=A(2)
           A(4)=A(1)
           NGAM=NT1/8192
           IF (NT1.LT.0) THEN
             NGAM=NGAM+7
           ELSE
           ENDIF
           NGAM=NGAM+1
           IF ((NGAM.NE.0).AND.(NGAM.LT.7)) GO TO
             NGAM=7
1            NT1=NT1*8
             IF (NT1.LT.0) THEN
              R=-4096-NT1/8
             ELSE
              R=NT1/8
             ENDIF
             R=R*FLOAT(IGAM(NGAM))
         RETURN
        END
```

```
 2      $FREEFORM
 3       CHARACTER*10 FICHE,FICHS,ACCE
 4       DIMENSION T(4096),T1(2048),T2(2048),T3(1024)
 5       EQUIVALENCE (T(1),T1(1)),(T(2049),T2(1))
 6         WRITE(6,61)
 7      61   FORMAT(' FE=',$)
 8         READ(5,51)FICHE
 9      51   FORMAT(A10)
10         WRITE(6,62)
11      62   FORMAT(' FS=',$)
12         READ(5,51)FICHS
13         OPEN(12,FILE=FICHE,STATUS='OLD',ACCESS=
        &       'SEQUENTIAL',FORM='UNFORMATTED')
14         READ(12)T
15         CALL FFT4KM(T1,T2,T3,2048)
16         CALL NORM(T,4096)
17         OPEN(16,FILE=FICHS,STATUS='NEW',ACCESS=
        &       'DIRECT',FORM='UNFORMATTED',RECL=4096)
18           J1=1
19           J2=1024
20           DO 10 I=1,4
21             WRITE(16,REC=I)(T(J),J=J1,J2)
22             J1=J1+1024
23      10     J2=J2+1024
24             CLOSE(16)
25             CLOSE(12)
26       STOP
27       END
```

```
 2      $FREEFORM
 3          SUBROUTINE NORM(TR,N)
 4            DIMENSION TR(1)
 5            VALMAX=0.
 6             DO 1 I=1,N
 7                 IF(TR(I).LE.VALMAX) GO TO 1
 8                    VALMAX=TR(I)
 9      1      CONTINUE
10             DO 3 I=1,N,2
11                 TR(I)=TR(I)/VALMAX
12      3          TR(I+1)=TR(I+1)/VALMAX
13        RETURN
14       END
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 00D5H     213D
     VARIABLE AREA SIZE = 000CH      12D
     MAXIMUM STACK SIZE = 0002H       2D
     14 LINES READ
```

```
      $FREEFORM
        SUBROUTINE FFT4KM(TR,TI,S,N)
         DIMENSION TR(1),TI(1),S(1)
          NS16=N/16
          NS8=NS16+NS16
          NS4=NS8+NS8
          NS2=NS4+NS4
          NS816=NS16+NS8
          RS=0.707107
          PI=3.14159226
          FN=N
          PISN=PI/FN
             TR0=TR(1)
             TR(1)=TR(2)
             TRNP1=-TI(N)
             CALL HERME1(TR,TR,S,1,NS2-1,0,1)
             S(NS2)=TI(1)
             TR(NS2+1)=(TI(2)-TR(N))/2.
             CALL HERME1(TR,TI,TI,NS2+2,N,NS2+1,0)

             J=NS2+2
             TI(NS2+2)=TI(1)
             DO 3 I=2,NS2-1
                J=J+1
                TI(J)=TI(I)
    3           TI(I)=S(I-1)
                TI(NS2)=S(NS2-1)
                TI(NS2+1)=S(NS2)
                TI(1)=TR0
                     C1=COS(PISN)
                     S1=SIN(PISN)
                     A=1.
                     B=0.
                     II=0
                     I=NS2+2
                     J=NS2
                     K=N
                     DO 4 L=2,NS4
                        IF (II.EQ.7) GO TO 5
                           II=II+1
                           CALL ANGLE(A,B,C1,S1)
                           GO TO 6
    5                    FN=L-1
                         FIC=PISN*FN
                         A=COS(FIC)
                         B=SIN(FIC)
                         II=0
    6                     CALL HERME2(TR(L),TR(K),TI(L),TI(K),
                          CALL HERME2(TR(J),TR(I),TI(J),TI(I),
                           J=J-1
                           I=I+1
    4                      K=K-1
                    CALL HERME2(TR(J),TR(K),TI(J),TI(K),0.70
```

```
                    A=TR(1)
                    TR(1)=A+TRNP1
                    TI(1)=A-TRNP1
                    TI(NS2+1)=-TI(NS2+1)*2.
                    TR(NS2+1)=TR(NS2+1)*2.
              S(NS4+1)=1.1
              S(NS4+2)=NS8+1.1
              II=NS4+3
              JJ=NS16
              K=2
    7         I=NS4+K
              DO 8 J=NS4+1,I
                 S(II)=S(J)+JJ
    8            II=II+1
            JJ=JJ/2
            K=K+K
            IF (K.LT.NS16) GO TO 7
            CALL TRIAG(TR,S,N)
            CALL TRIAG(TI,S,N)

          FN=NS2
          PIC=PI/FN
          DO 40 I=1,3
             FN=I
             FIC=PIC*FN
             S(I)=COS(FIC)
    40       S(NS4-I)=SIN(FIC)
             J=NS4
             DO 50 I=4,NS8-4,4
                FN=I
                FIC=PIC*FN
                C1=COS(FIC)
                S1=SIN(FIC)
                S(I)=C1
                J=J-4
                S(J)=S1
                DO 50 L=1,3
                S(I+L)=C1*S(L)-S1*S(NS4-L)
    50          S(J-L)=S1*S(L)+C1*S(NS4-L)
                S(NS8)=RS
                S(NS4)=0.
```

```
                I=1
60               CALL COOLEY(TR(I),TI(I),TR(I+1),TI(I+1))
                 I=I+2
                IF (I.LT.N) GO TO 60
                 I=1
70               CALL COOLEY(TR(I),TI(I),TR(I+2),TI(I+2))
                 I=I+1
                 CALL COOLE1(TR(I),TI(I),TR(I+2),TI(I+2))
                 I=I+3
                IF (I.LT.N) GO TO 70
                 I=1
80               CALL COOLEY(TR(I),TI(I),TR(I+4),TI(I+4))
                 I=I+1
                 CALL ANGLE(TR(I+4),TI(I+4),RS,RS)
                 CALL COOLEY(TR(I),TI(I),TR(I+4),TI(I+4))
                 I=I+1
                 CALL COOLE1(TR(I),TI(I),TR(I+4),TI(I+4))
                 I=I+1
              C1=TR(I+4)*RS
              S1=TI(I+4)*RS
              TR(I+4)=TR(I)+C1+S1
              TI(I+4)=TI(I)-C1+S1
              TR(I)=TR(I)-C1-S1
              TI(I)=TI(I)+C1-S1
              I=I+5
              IF (I.LT.N) GO TO 80


                 IDW=NS16
                 K=4
90               L=1
91                  I=IDW
                    II=L+K
                     J=II+K
                    JJ=J+K
                    CALL COOLEY(TR(L),TI(L),TR(J),TI(J))
                    CALL COOLE1(TR(II),TI(II),TR(JJ),TI(JJ))
92                        L=L+1
                          J=J+1
                          A=S(I)
                          B=S(NS4-I)
                          CALL ANGLE(TR(J),TI(J),A,B)
                          CALL COOLEY(TR(L),TI(L),TR(J),TI(
                          II=II+1
                          JJ=JJ+1
                          C1=TR(JJ)*B+TI(JJ)*A
                          S1=TR(JJ)*A-TI(JJ)*B
                          TR(JJ)=TR(II)+C1
                          TI(JJ)=TI(II)-S1
                          TR(II)=TR(II)-C1
                          TI(II)=TI(II)+S1
                          I=I+IDW
                          IF (I.LT.NS4) GO TO 92
                    J=JJ+1
                    I   (..T.,N  GO TO 9
                 IDW=IDW/2
                K=K+K
                   . T (K2+1) GO  O ,0
```

```
156              PISN=PISN/2.
157              C1=COS(PISN)
158              S1=SIN(PISN)
159              A=-C1
160              B=-S1
161              CALL COOLE4(TI(1),TI(N),B)
162              TR(1)=0.
163              TR(NS2+1)=-TR(NS2+1)*2.
164              L=1
165              I=N
166              J=1
167       100      J=J+1
168                CALL ANGLE(A,B,C1,S1)
169                CALL COOLE4(TR(J),TR(I),B)
170                CALL ANGLE(A,B,C1,S1)
171                I=I-1
172                CALL COOLE4(TI(J),TI(I),B)
173                IF (J.EQ.NS2) GO TO 120
174                   J=J+1
175                   A=-S(L)
176                   B=-S(NS4-L)
177                   L=L+1
178                   CALL COOLE4(TR(J),TR(I),B)
179                   CALL ANGLE(A,B,C1,S1)
180                   I=I-1
181                   CALL COOLE4(TI(J),TI(I),B)
182                   GO TO 100

184       120     DO 130 I=1,NS2
185       130        S(I)=TI(I)
186               J=NS2+1
187               DO 140 I=2,N,2
188                  TI(I)=TI(J)
189                  TI(I-1)=TR(J)
190       140        J=J+1
191               J=NS2
192               I=N
193       150     TR(I)=S(J)
194               TR(I-1)=TR(J)
195               J=J-1
196               I=I-2
197               IF (I.GT.1) GO TO 150
198        RETURN
199        END
```

MODULE INFORMATION:

```
     CODE AREA SIZE     = 109AH    4250D
     VARIABLE AREA SIZE = 005CH      92D
     MAXIMUM STACK SIZE = 0006H       6D
     199 LINES READ
```

```
 2      $FREEFORM
 3        SUBROUTINE TRIAG(TA,SS,N)
 4         DIMENSION TA(1),SS(1)
 5              NS16=N/16
 6              NS8=NS16+NS16
 7              NS4=NS8+NS8
 8              NS2=NS4+NS4
 9              NS816=NS8+NS16
10              II=0
11              JJ=0
12      9       JJ=JJ+1
13              IF (JJ-2) 10,11,12
14      10         J=N-3
15                 L=NS4
16                 I=J
17      13           SS(L)=TA(J)
18                   L=L-1
19                   TA(I)=TA(J-1)
20                   TA(I-1)=TA(J-2)
21                   TA(I-2)=TA(J-3)
22                   I=I-3
23                   J=J-4
24                 IF (L.NE.1) GO TO 13
25                   SS(1)=TA(1)
26                   GO TO 20
27      11         J=N-1
28      $EJECT
```

```
 29                L=NS
 30                I=J
 31      14           SS(L)=TA(J)
 32                   L=L-1
 33                   TA(I)=TA(J-1)
 34                   TA(I-1)=TA(J-2)
 35                   I=I-2
 36                   J=J-3
 37                   IF (L.NE.1) GO TO 14
 38                   SS(1)=TA(NS4+2)
 39                   TA(NS2+1)=TA(NS4+1)
 40                   GO TO 20
 41      12        IF (JJ.EQ.4) GO TO 18
 42                   J=N-1
 43                   L=NS4
 44                   I=J
 45      15              SS(L)=TA(J)
 46                      L=L-1
 47                      TA(I)=TA(J-1)
 48                      J=J-2
 49                      I=I-1
 50                      IF (L.NE.1) GO TO 15
 51                      SS(1)=TA(NS2+1)
 52                      GO TO 20
 53      18        J=NS2+NS4+1
 54                DO 17 I=1,NS4
 55                   SS(I)=TA(J)
 56      17           J=J+1
 57      20        DO 19 L=1,NS16
 58                   K=SS(L+NS4)+II
 59                   TA(K)=SS(L)
 60                   TA(K+1)=SS(L+NS8)
 61                   TA(K+2)=SS(L+NS16)
 62      19           TA(K+3)=SS(L+NS816)
 63                   II=II+NS4
 64                   IF (JJ.NE.4) GO TO 9
 65       RETURN
 66       END
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 0413H    1043D
     VARIABLE AREA SIZE = 001EH      30D
     MAXIMUM STACK SIZE = 0004H       4D
     66 LINES READ

     0 PROGRAM ERROR(S) IN PROGRAM UNIT TRIAG

     0 TOTAL PROGRAM ERROR(S)
     END OF FORTRAN COMPILATION
```

```
 2      $FREEFORM
 3        SUBROUTINE COOLEY(U,V,W,X)
 4          Y=W
 5          Z=X
 6          W=U-Y
 7          X=V-Z
 8          U=U+Y
 9          V=V+Z
10        RETURN
11       END
```

```
 2      $FREEFORM
 3        SUBROUTINE COOLE1(U,V,W,X)
 4          Y=X
 5          Z=W
 6          W=U+Y
 7          X=V-Z
 8          U=U-Y
 9          V=V+Z
10        RETURN
11       END
```

```
 2      $FREEFORM
 3        SUBROUTINE COOLE4(U,V,W)
 4          Y=U-V
 5          Z=(U+V)/W
 6          U=Z-Y
 7          V=Z+Y
 8        RETURN
 9       END
```

```
 2      $FREEFORM
 3        SUBROUTINE ANGLE(U,V,W,X)
 4          Y=U
 5          U=U*W-V*X
 6          V=V*W+Y*X
 7        RETURN
 8       END
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 0086H     134D
     VARIABLE AREA SIZE = 000CH      12D
     MAXIMUM STACK SIZE = 000AH      10D
     8 LINES READ
```

```
  2      $FREEFORM
  3        SUBROUTINE HERME1(TA,TB,TC,N1,N2,M1,M2)
  4          DIMENSION TA(1),TB(1),TC(1)
  5            J=1
  6            DO 1 I=N1,N2
  7               J=J+2
  8               TC(I-M1)=TB(J)
  9      1        TA(I+M2)=(TB(J+1)-TB(J-1))/2.
 10        RETURN
 11       END
```

```
  2      $FREEFORM
  3        SUBROUTINE HERME2(U,V,W,X,Y,Z)
  4          A1=U+V
  5          A6=U-V
  6          A5=W+X
  7          A4=W-X
  8            A2=-Z*A6-Y*A5
  9            A3=Y*A6-Z*A5
 10            U=A1+A2
 11            V=A1-A2
 12            W=A4+A3
 13            X=A3-A4
 14        RETURN
 15       END
```

```
MODULE INFORMATION:

     CODE AREA SIZE      = 0126H     294D
     VARIABLE AREA SIZE  = 0024H      36D
     MAXIMUM STACK SIZE  = 000AH      10D
     15 LINES READ
```

```
      $FREEFORM
        CHARACTER*10 FICHE,FICHS
        DIMENSION AA(109),BB(201),G(5120),
     &     B(1044),T(1024)
        EQUIVALENCE (B(21),T(1)),(BB(1),G(1))
          WRITE(6,62)
62         FORMAT('FE=',$)
          READ(5,52)FICHE
52         FORMAT(A10)
          WRITE(6,63)
63         FORMAT('FS=')
          READ(5,52)FICHS
          OPEN(16,FILE=FICHE,ACCESS='DIRECT',
     &          FORM='UNFORMATTED',STATUS='OLD',
     &          RECL=4096)
          OPEN(17,FILE=FICHS,ACCESS='DIRECT',
     &          FORM='UNFORMATTED',STATUS='NEW',
     &          RECL=20480)
          WRITE(6,61)
61         FORMAT('FENETRE=',$)
          READ(5,51)IAP
51         FORMAT(I2)
          CALL FILTRE(201,IAP,AA,BB)

5         WRITE(6,60)
60         FORMAT('NUBL=',$)
          READ(5,51)NB
          IF (NB.GT.4) GO TO 4
           IF (NB.EQ.1) THEN
                DO 1 I=1,20
1                B(I)=0
              READ(16,REC=1)T
              CALL INTER3(G,B,AA)
              WRITE(17,REC=NB)G
          ELSE
              I=NB-1
              READ(16,REC=I)T
              DO 2 I=1,20
2                B(I)=T(I+1004)
              READ(16,REC=NB)T
              CALL INTER3(G,B,AA)
              WRITE(17,REC=NB)G
           ENDIF
          GO TO 5
4         CLOSE(17)
          CLOSE(16)
        STOP
       END
```

```
      $FREEFORM
        SUBROUTINE FILTRE(NP,IF,A,B)
        DIMENSION B(1),A(1)
          DO 1 I=2,200
    1        B(I)=1.
          B(201)=0.5
          IF (IF.EQ.0) THEN
             MP=201
             GO TO 8
          ELSE
             MP=NP
          ENDIF
          IF (IF-2)3,4,5
    3       CALL FENETR(B,MP,0.,0.,1.,2.,0.)
            GO TO 8
    4       CALL FENETR(B,MP,0.06267,0.33213,
     &                    0.21839,2.,0.38680)
            GO TO 8
    5     IF (IF-3)6,6,7
    6       CALL FENETR(B,MP,0.23978,0.45808,
     &                    0.22499,2.,0.077208)
            GO TO 8
    7       CALL FENETR(B,MP,0.,0.,1.,2.5,0.)
    8     CALL FENETO(A,B,MP)
        RETURN
        END


      $FREEFORM
        SUBROUTINE FENETR(TR,M,R,S,T,V,W)
         DIMENSION TR(1)
          DO 1 I=2,M
            FM=M-1
            FI=I-1
            F=1 -(FI*FI)/(FM*FM)
    1   TR(I)=TR(I)*(R+S*F+T*F**V+W*F**3)
        RETURN
        END


      $FREEFORM
        SUBROUTINE FENETO(TR,TI,M)
        DIMENSION TR(1),TI(1)
           AMAX=0.5
            DO 11 I=2,M
   11          AMAX=AMAX+TI(I)
            SIA=+0.3141592E-02
            DO 12 I=56,105
               FI=I-55
               SIG=SIA*FI
               TR(I)=0.5
                DO 13 J=2,M
                   FJ=J-1
   13              TR(I)=TR(I)+TI(J)*COS(SIG*FJ)
   12       TR(I)=(TR(I)/AMAX)*(1.-(FI*FI)/2500.)**2
            DO 14 I=106,109
               TR(I)=0.
            DO 15 I=
```

```
 2      $FREEFORM
 3        SUBROUTINE INTER3(TR,TI,A)
 4         DIMENSION TR(1),TI(1),A(1)
 5           DO 1 K=1,5120
 6              TR(K)=0.0
 7              I=(K-1)/5
 8              J=I*5-K+1
 9               DO 1 N=1,21
10                  I=I+1
11                  J=J+5
12      1       TR(K)=TR(K)+TI(I)*A(J)
13          RETURN
14       END
```

MODULE INFORMATION:

```
CODE AREA SIZE      = 00CDH     205D
VARIABLE AREA SIZE  = 000EH      14D
MAXIMUM STACK SIZE  = 0004H       4D
14 LINES READ

0 PROGRAM ERROR(S) IN PROGRAM UNIT INTER3

0 TOTAL PROGRAM ERROR(S)
END OF FORTRAN COMPILATION
```

```
      $FREEFORM
        DIMENSION TR(5120)
        CHARACTER FICHE*10
              WRITE(6,60)
      60       FORMAT('FE=',$)
              READ(5,50)FICHE
      50       FORMAT(A10)
              OPEN(16,FILE=FICHE,ACCESS='DIRECT',
      &            STATUS='OLD',FORM='UNFORMATTED',
      &            RECL=20480)
      6          WRITE(6,61)
      61          FORMAT('NUBL=',$)
                 READ(5,51)NB
      51          FORMAT(I2)
                 IF (NB.LT.4) THEN
                    READ(16,REC=NB)TR
                    CALL TRACE2(TR,5120)
                    GO TO 6
                 ELSE
                 ENDIF
         CLOSE(16)
        STOP
       END
```

```
      $FREEFORM
       SUBROUTINE TRACE2(TR,NSORT)
        DIMENSION TR(1),ORD1(11),ORD2(11)
        CHARACTER IMP
        DATA ORDMAX/0./,ORDMIN/0./
         DO 1 J=1,NSORT
          IF(TR(J).LT.ORDMIN)ORDMIN=TR(J)
      1   IF(TR(J).GT.ORDMAX)ORDMAX=TR(J)
          ECHEL1=(ORDMAX-ORDMIN)/100.
          ECHEL2=(ORDMAX-ORDMIN)/70.
         IF(ECHEL1.LT.0.1E-30) RETURN
          ORD1(1)=ORDMIN
          ORD2(1)=ORDMIN
         DO 3 K=1,9
           ORD2(K+1)=ORD2(K)+ECHEL2*7.
      3    ORD1(K+1)=ORD1(K)+ECHEL1*100.
           ORD1(11)=ORDMAX
           ORD2(11)=ORDMAX
      $EJECT
```

```
      200 WRITE(6,100)
      100  FORMAT(':LP:?',$)
          READ(5,101)IMP
      101  FORMAT(A1)
          IF (IMP.NE.'E') THEN
           WRITE(6,110)
      110   FORMAT('DB=',$)
           READ(5,111)NSORT1
      111   FORMAT(I4)
           WRITE(6,112)
      112   FORMAT('FN=',$)
           READ(5,111)NSORT2
           IF (IMP.EQ.'Y') THEN
            NU=7
            OPEN(NU,FILE=':LP:',STATUS='NEW',
     &      ACCESS='SEQUENTIAL',FORM='FORMATTED')
            WRITE(NU,103)ORD1,ECHEL1
      103    FORMAT(10X,11(3X,F7.4),'EC=',F6.5)

          WRITE(NU,104)
      104 FORMAT(2X,15('-'),'+',10('---------+'),'-->')
          CALL SPECTR(TR,NSORT1,NSORT2,ECHEL1,NU,ORDMIN)
          CLOSE(NU)
          GO TO 200
         ELSE
          IF (IMP.EQ.'D') THEN
           NU=10
           OPEN(NU,FILE=':F0:CONSER.TMP',ACCESS=
     &          'SEQUENTIAL',
     &           STATUS='NEW',FORM='FORMATTED')
          ELSE
           NU=6
          END IF
          WRITE(NU,102)ORD2
      102  FORMAT(11(2X,F5.2))
          WRITE(NU,107)
      107  FORMAT(11('----+--'),'-->')
          CALL SPECTR(TR,NSORT1,NSORT2,ECHEL2,NU,ORDMIN)
          GO TO 200
          END IF
          ELSE
          END IF
          IF (NU.EQ.10) THEN
            CLOSE(10)
          ELSE
          END IF
        RETURN
      END
```

```
MODULE INFORMATION:

     CODE AREA SIZE      = 04F2H    1266D
     VARIABLE AREA SIZE  = 009DH     157D
     MAXIMUM STACK SIZE  = 000AH      10D
     68 LINES READ
```

```
 2      $FREEFORM
 3        SUBROUTINE SPECTR(TA,N1,N2,Y,NOUT,Y2)
 4          DIMENSION TA(1),SORT(101),SORT1(79)
 5          EQUIVALENCE (SORT(1),SORT1(1))
 6          DATA ASTER/'*'/,BLANC/' '/,
       &          SORT/101*' '/,SORT1/79*' '/,Y3/1.5/
 7               DO 4 I=N1,N2,4
 8                   J1=(TA(I)-Y2)/Y+Y3
 9                   J2=(TA(I+1)-Y2)/Y+Y3
10                   J3=(TA(I+2)-Y2)/Y+Y3
11                   J4=(TA(I+3)-Y2)/Y+Y3
12                      SORT(J1)=ASTER
13                      SORT(J2)=ASTER
14                      SORT(J3)=ASTER
15                      SORT(J4)=ASTER
16                   IF (NOUT.EQ.6) THEN
17                      WRITE(NOUT,109)SORT1
18      109              FORMAT(79A1)
19                   ELSE
20                     FI=I
21                      WRITE(NOUT,105)FI,SORT
22      105              FORMAT(1X,F11.3,2X,'!',2X,101A1)
23                    END IF
24                      SORT(J1)=BLANC
25                      SORT(J2)=BLANC
26                      SORT(J3)=BLANC
27      4               SORT(J4)=BLANC
28          RETURN
29        END
```

```
MODULE INFORMATION:

     CODE AREA SIZE     = 022AH     554D
     VARIABLE AREA SIZE = 01CEH     462D
     MAXIMUM STACK SIZE = 0008H       8D
     30 LINES READ

     0 PROGRAM ERROR(S) IN PROGRAM UNIT SPECTR

     0 TOTAL PROGRM ERROR(S)
     END OF FORTRAN COMPILATION
```

```
                $EJECT
  1             V9600:DO;
  2    1         V9600:PROCEDURE PUBLIC;
  3    2                OUTPUT(0F5H) = 40H;
  4    2                OUTPUT(0F3H) = 036H;
  5    2                OUTPUT(0F0H) = LOW(0008H);
  6    2                OUTPUT(0F0H) = HIGH(0008H);
  7    2                OUTPUT(0F5H) = 08EH;
  8    2                OUTPUT(0F5H) = 025H;
  9    2         END V9600;
 10    1        END V9600;


MODULE INFORMATION:

      CODE AREA SIZE      = 0019H      25D
      VARIABLE AREA SIZE  = 0000H       0D
      MAXIMUM STACK SIZE  = 0000H       0D
      11 LINES READ
      0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
            $EJECT
1           SBC80:DO;
            /*****************************************
            *                                       *
            *      SBC80.SRC                        *
            *                                       *
            * PROCEDURES PUBLIQUES: INIT            *
            *                       INTERRUPTION    *
            *                       SEND            *
            *                       RECEIVE         *
            *                       ARRET$SPECTRO   *
            *                       DEPART$SPECTRO  *
            *                       SPECTRO$PRET    *
            *                       FIN$TRANSMISSION*
            * PROCEDURES EXTERNES: PUT              *
            *****************************************/


            $EJECT
2   1       DECLARE DCL   LITERALLY 'DECLARE',
                    EQU   LITERALLY 'LITERALLY',
                    PROC  EQU   'PROCEDURE',
                    TRUE        EQU   '0FFH',
                    FALSE       EQU   '0',
                    PPI1        EQU   '0E7H',
                    PPI2        EQU   '0EBH',
                    PORT3       EQU   '0E6H',
                    PORT6       EQU   '0EAH',
                    EN$INTE     EQU   '00001001B',
                    INTR        EQU   '00001000B',
                    PC6PPI1     EQU   '01000000B',
                    PC6PI2      EQU   '01000000B',
                    PC7PPI2     EQU   '10000000B',
                    OCT1        EQU   'INPUT(0E8H)',
                    OCT2        EQU   'INPUT(0E4H)',
                    OCT3        EQU   'INPUT(0E5H)',
                    OCT4        EQU   'INPUT(0E9H)',
             SPPR       EQU '(INPUT(PORT3) AND PC6PPI1)=0',
             USART$IN   EQU '(INPUT(USART) AND 2) <> 0',
             USART$OUT  EQU '(INPUT(USART) AND 1) <> 0',
             PPI$INT    EQU '(INPUT(PORT3) AND INTR) <> 0',
             SPSTO      EQU 'INPUT(PORT6) OR PC6PPI2',
            DEFAUT      EQU 'INPUT(PORT6) OR PC7PPI2',
             NSPSTO     EQU 'INPUT(PORT6) AND NOT PC6PPI2',
             NDEFAUT    EQU 'INPUT(PORT6) AND NOT PC7PPI2',
                    USART       EQU   '0EDH',
                    LIGNE       EQU   '0ECH',
                    MODE$UN     EQU   '0BAH',
                    MODE$ZERO   EQU   '093H',
                    MODE$8251   EQU   '08EH',
                    COMANDE     EQU   '27H';
```

```
                $EJECT
 3    1         PUT:
                  PROCEDURE EXTERNAL;
 4    2         END PUT;
 5    1         DECLARE MESURE(4) BYTE EXTERNAL;
 6    1         INIT:
                PROCEDURE PUBLIC;
 7    2             OUTPUT(USART) = MODE$8251;
 8    2             OUTPUT(USART) = COMMANDE;
 9    2             OUTPUT(PPI1)  = MODE$UN;
10    2             OUTPUT(PPI1)  = EN$INTE;
11    2             OUTPUT(PPI2)  = MODE$ZERO;
12    2         END INIT;
13    1         SEND:
                  PROCEDURE(CAR) PUBLIC;
14    2           DECLARE CAR BYTE;
15    2           DO WHILE NOT USART$OUT;
16    3           END;
17    2           OUTPUT(LIGNE) = CAR;
18    2         END SEND;
19    1         RECEIVE:
                  PROCEDURE BYTE PUBLIC;
20    2           DO WHILE NOT USART$IN;
21    3           END;
22    2           RETURN INPUT(LIGNE);
23    2         END RECEIVE;
24    1         ARRET$SPECTRO:
                  PROCEDURE PUBLIC;
25    2           OUTPUT(PORT6) = SPSTO;
26    2         END ARRET$SPECTRO;
27    1         DEPART$SPECTRO:
                  PROCEDURE PUBLIC;
28    2           OUTPUT(PORT6) = NSPSTO;
29    2         END DEPART$SPECTRO;
```

```
                $EJECT
  30    1       ACQUISITION:
                  PROCEDURE BYTE;
  31    2           IF PPI$INT
                     THEN
  32    2             DO;
  33    3               MESURE(0) = OCT1;
  34    3               MESURE(2) = OCT3;
  35    3               MESURE(3) = OCT4;
  36    3               MESURE(1) = OCT2;
  37    3               RETURN TRUE;
  38    3             END;
                    ELSE
  39    2             DO;
  40    3               RETURN FALSE;
  41    3             END;
  42    2       END ACQUISITION;
  43    1       INTERRUPTION:
                  PROCEDURE  PUBLIC;
  44    2         DISABLE;
  45    2         IF ACQUISITION = TRUE
                    THEN
  46    2           DO;
  47    3             CALL PUT;
  48    3           END;
                   /* SUPPRIMMER LA CAUSE D'INTERRUPT */
  49    2       END INTERRUPTION;
  50    1       FIN$TRANSMISSION:
                  PROCEDURE PUBLIC;
  51    2         CALL DEPART$SPECTRO;
  52    2         HALT;
  53    2       END FIN$TRANSMISSION;
  54    1       SPECTRO$PRET:
                  PROCEDURE BYTE PUBLIC;
  55    2       RETURN SPPR;
  56    2       END SPECTRO$PRET;
  57    1       END SBC80;


MODULE INFORMATION:

     CODE AREA SIZE      = 008CH    140D
     VARIABLE AREA SIZE  = 0001H      1D
     MAXIMUM STACK SIZE  = 0002H      2D
     120 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
                $EJECT
  1             MICHEL:DO;
                /*************************************************
                *                                                *
                *              MICHEL.SRC                        *
                *                                                *
                * PROCEDURES                    PROCEDURES       *
                * EXTERNES:  INIT               PUBLIQUES:  PUT  *
                *            INTERRUPTION                        *
                *            SEND                                *
                *            RECEIVE                             *
                *            EMISSION                            *
                *            ECHO                                *
                *            ARRET$SPECTRO                       *
                *            DEPART$SPECTRO                      *
                *            SPECTRO$PRET                        *
                *            FIN$TRANSMISSION                    *
                *                                                *
                *************************************************/


                $EJECT
  2     1       DECLARE DCL      LITERALLY 'DECLARE',
                        EQU      LITERALLY 'LITERALLY',
                        PROC     EQU  'PROCEDURE',
                        TRUE     EQU  '0FFH',
                        FALSE    EQU  '0',
                        STX      EQU  '02H',
                        ETX      EQU  '03H',
                        SPPR     EQU  '11H',
                        SPNPR    EQU  '23H',
                        TNT      EQU  '980D';
  3     1       DECLARE MESURE(4) BYTE PUBLIC;
  4     1       DECLARE ENVOI(4) BYTE PUBLIC;
  5     1       DECLARE NT(TNT) BYTE ;
  6     1       DECLARE PTR$GET ADDRESS;
  7     1       DECLARE PTR$PUT ADDRESS;
  8     1       DECLARE LG$NT ADDRESS;
  9     1       INIT:
                  PROCEDURE EXTERNAL;
 10     2       END INIT;
 11     1       ACK:
                  PROCEDURE BYTE EXTERNAL;
 12     2       END ACK;
 13     1       NAK:
                  PROCEDURE BYTE EXTERNAL;
 14     2       END NAK;
 15     1       SEND:
                  PROCEDURE (CAR) EXTERNAL;
 16     2          DECLARE CAR BYTE;
 17     2       END SEND;
 18     1       RECEIVE:
                  PROCEDURE  BYTE EXTERNAL;
 19     2       END RECEIVE;
 20     1       EMISSION:
                  PROCEDURE EXTERNAL;
 21     2       END EMISSION;
```

```
                $EJECT
 22    1        ECHO:
                  PROCEDURE BYTE EXTERNAL;
 23    2        END ECHO;
 24    1        ARRET$SPECTRO:
                  PROCEDURE EXTERNAL;
 25    2        END ARRET$SPECTRO;
 26    1        DEPART$SPECTRO:
                  PROCEDURE EXTERNAL;
 27    2        END DEPART$SPECTRO;
 28    1        SPECTRO$PRET:
                  PROCEDURE BYTE EXTERNAL;
 29    2        END SPETRO$PRET;
 30    1        INTERRUPTION:
                  PROCEDURE EXTERNAL;
 31    2        END INTERRUPTION;
 32    1        FIN$TRANSMISSION:
                  PROCEDURE EXTERNAL;
 33    2        END FIN$TRANSMISSION;
 34    1        I7:
                  PROCEDURE INTERRUPT 7;
 35    2          DISABLE;
 36    2          CALL INTERRUPTION;
 37    2        END I7;

                $EJECT
 38    1        PUT:
                  PROCEDURE PUBLIC;
 39    2          DCL I BYTE;
 40    2          DO I = 0 TO 3;
 41    3            NT(PTR$PUT) = MESURE(I);
 42    3            PTR$PUT = PTR$PUT + 1;
 43    3            IF PTR$PUT = TNT
                       THEN
 44    3                DO; PTR$PUT = 0; END;
 47    3            END;
 48    2            LGNT = LG$NT + 4;
 49    2            IF LG$NT = TNT      /* NT PLEIN ? */
                       THEN
 50    2                DO; CALL ARRET$SPECTRO; END;
 53    2        END PUT;
 54    1        GET: PROC BYTE;
 55    2          DECLARE I BYTE;
 56    2          IF LG$NT = 0      /* NT VIDE ? */
                       THEN
 57    2                DO; RETURN FALSE; END;
 60    2          DO I = 0 TO 3;
 61    3            ENVOI(I) = NT(PTR$GET);
 62    3            PTR$GET = PTR$GET + 1;
 63    3            IF PTR$GET = TNT
                       THEN
 64    3                DO; PTR$GET = 0; END;
 67    3            END;
 68    2            LG$NT = LG$NT - 4;
 69    2            IF LG$NT  = TNT - 4    /* LE NT ETAIT PLEIN ? */
                       THEN
 70    2                DO;
 71    3                  CALL  DEPART$SPECTRO;
 72    3                END;
 73    2            RETURN TRUE;
 74    2        END GET;
```

```
                $EJECT
  75    1       DECLARE ECHO1 BYTE;
  76    1       DECLARE CAR BYTE;
  77    1       DECLARE MAL$RECU BYTE;
  78    1           DISABLE;
  79    1           PTR$PUT = 0;
  80    1           PTR$GET = 0;
  81    1           LG$NT = 0;
  82    1           CALL INIT;
  83    1           CALL ARRET$SPECTRO;
  84    1           CAR = RECEIVE AND 07FH;
  85    1           DO WHILE NOT SPECTRO$PRET;
  86    2           CALL SEND(SPNPR);
  87    2           CAR = RECEIVE AND 07FH;
  88    2           END;
  89    1           CALL DEPART$SPECTRO;
  90    1           MAL$RECU = TRUE;
  91    1           DO WHILE MAL$RECU;
  92    2             CALL SEND(SPPR);
  93    2             CAR = RECEIVE AND 07FH;
  94    2             IF CAR = SPPR
                       THEN
  95    2                 DO; MAL$RECU = FALSE; END;
  98    2           END;
  99    1           ENABLE;
 100    1           DO WHILE TRUE;
 101    2             DO WHILE GET = FALSE;/* TANT QUE NT VIDE */
 102    3             END;
 103    2             ECHO1 = FALSE;
 104    2             CAR = ACK;
 105    2             DO WHILE ECHO1 = FALSE;
 106    3                CALL SEND(CAR);
 107    3                CAR = RECEIVE AND 07FH;
 108    3                IF CAR = ETX
                           THEN
 109    3                   DO;
 110    4                     CALL FIN$TRANSMISSION;
 111    4                   END;
 112    3                CALL EMISSION;
 113    3                ECHO1 = ECHO;
 114    3                CAR = NAK;
 115    3             END;
 116    2          END;
 117    1       END MICHEL;


MODULE INFORMATION:

     CODE AREA SIZE     = 0183H    387D
     VARIABLE AREA SIZE = 03E7H    999D
     MAXIMUM STACK SIZE = 000AH     10D
     159 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

```
                $EJECT
  1             MDS221:DO;
                /*************************************************
                *                                                *
                *      MDS221.SRC                                *
                *                                                *
                * PROCEDURES PUBLIQUES:  PROCEDURES EXTERNES:    *
                * EMISSION               SEND                    *
                * ECHO                   RECEIVE                 *
                *                                                *
                *************************************************/


                $EJECT
  2     1       DECLARE LIT LITERALLY 'LITERALLY',
                        DCL LIT 'DECLARE',
                        PROC LIT 'PROCEDURE',
                        TRUE LIT '0FFH',
                        FALSE LIT '0';
  3     1       SEND: PROC(CAR) EXTERNAL;
  4     2         DCL CAR BYTE;
  5     2       END SEND;
  6     1       RECEIVE: PROC BYTE EXTERNAL;
  7     2         DCL CAR BYTE;
  8     2       END RECEIVE;
  9     1       NAK:
                  PROCEDURE BYTE PUBLIC;
 10     2         RETURN 15H;
 11     2       END NAK;
 12     1       ACK:
                  PROCEDURE BYTE PUBLIC;
 13     2       RETURN 06H;
 14     2       END ACK;
 15     1       DECLARE ENVOI(4) BYTE EXTERNAL;
 16     1       DECLARE RETOUR(4) BYTE;


                $EJECT
 17     1       EMISSION:
                PROCEDURE PUBLIC;
 18     2         CALL SEND(ENVOI(0));
 19     2         RETOUR(0) = RECEIVE;
 20     2         CALL SEND(ENVOI(1));
 21     2         RETOUR(1) = RECEIVE;
 22     2         CALL SEND(ENVOI(2));
 23     2         RETOUR(2) = RECEIVE;
 24     2         CALL SEND(ENVOI(3));
 25     2         RETOUR(3) = RECEIVE;
 26     2       END EMISSION;
 27     1       ECHO:
                PROCEDURE BYTE PUBLIC;
 28     2       DECLARE I BYTE;
 29     2       DO I = 0 TO 3;
 30     3         IF RETOUR(I) <> ENVOI(I)
                    THEN
 31     3             DO;
 32     4               RETURN FALSE;
 33     4             END;
 34     3       END;
 35     2       RETURN TRUE;
 36     2       END ECHO;
 37     1       END MDS221;
```

```
                $EJECT
  1             FFT:DO;
  2    1        DECLARE LIT LITERALLY 'LITERALLY',
                DCL  LIT 'DECLARE',
                PROC LIT 'PROCEDURE',
                EXT  LIT 'EXTERNAL',ADDR LIT 'ADDRESS',
                (CAR,CAR1) BYTE,
                (NBR$RECU,NB$MESURES)  ADDR,
                (TONUM,FICH$RES$NUM,FICH$DEC$NUM) ADDR,
                (TINUM,CTR,LG,BID,CODEREP) ADDR,
                NB$MES$PTR ADDR,
                NB$MESURES$VOULU BASED NB$MES$PTR ADDR,
                NT(1) BYTE,
                A$NT ADDR,FE BYTE,
                RECU BASED A$NT BYTE,
                COMPTEUR BYTE,
                (I,J,NON$TROUVE,LIAISON) BYTE,
                (ECRITURE,NB$ERREURS) BYTE,
                NOMBRE$MESURES ADDR,
                NOM$FICHIER$MESURES(10)  BYTE,
                ETX LIT '03H',          ACK LIT '06H',
                NAK LIT '15H',          STX LIT '02H',
                SPPR LIT '11H',         SPNPR LIT '23H',
                TRUE LIT '0FFH',        FALSE LIT '0',
                RC LIT '0DH',           LF LIT '0AH',
                CO LIT '0',             CI LIT '1';

                $EJECT
  3    1        V9600: PROC EXT;END V9600;
  5    1        ERROR: PROC(ERRNUM) EXT;
  6    2           DCL  ERRNUM ADDR;END ERROR;
  8    1        EXIT: PROC EXT;END EXIT;
 10    1        WRITE: PROC(AF,BUF,CNT,STS) EXT;
 11    2            DCL (AF,BUF,CNT,STS) ADDR;
 12    2        END WRITE;
 13    1        READ: PROC(AF,BUF,CNT,ACT,STS) EXT;
 14    2           DCL (AF,BUF,CNT,ACT,STS) ADDR;
 15    2        END READ;
 16    1        CLOSE: PROC(AFTN,STATUS)EXT;
 17    2          DCL (AFTN,STATUS) ADDR;
 18    2        END CLOSE;
 19    1        OPEN: PROC(AF,FE,AC,MODE,STS) EXT;
 20    2          DCL (AF,FE,AC,MODE,STS) ADDR;
 21    2        END OPEN;
 22    1        FQOGO: PROC EXT;END FQOGO;
 24    1        NBMES:PROC(PARAM1) EXT;
 25    2         DCL PARAM1 ADDR;
 26    2        END NBMES;
```

```
                $EJECT
27    1         TTYIN: PROC BYTE;
28    2           DCL (CAR1,CAR2) BYTE;
29    2           PE = FLSE;
30    2           DO WHILE TRUE;
31    3             CAR1 = INPUT(0F5H);
32    3             IF (CAR1 AND 02H) <> 0
                      THEN
33    3               DO; IF (CAR1 AND 08H) <> 0
                          THEN DO; PE = TRUE; END;
38    4                 RETURN INPUT(0F4H);
39    4               END;
40    3           END;
41    2         END TTYIN;
42    1         FIN: PROC;
43    2           CALL WRITE(TONUM,.CAR,1,.CODEREP);
                    /* RENVOI DU DERNIER NUMERO */
44    2           CAR = TTYIN;
                              /* LECTURE 1 */
45    2           CALL WRITE(TONUM,.CAR,1,.CODEREP);
46    2           CAR = TTYIN;  /* LECTURE 2*/
47    2           CALL WRITE(TONUM,.CAR,1,.CODEREP);
48    2           CAR = TTYIN;
                  /* LECTURE OCTET TERMINAL */
49    2           CALL WRITE(TONUM,.CAR,1,.CODEREP);
50    2           CAR = TTYIN;
51    2           CALL WRITE(TONUM,.(ETX),1,.CODEREP);
52    2           CALL WRITE(FICH$RES$NUM,.NT,
                              NBR$RECU*2,.CODEREP);
53    2           CALL CLOSE(FICH$RES$NUM,.CODEREP);
54    2           CALL EXIT;
55    2         END FIN;
56    1         ECRITURE$DISQUE: PROC;
57    2           CALL WRITE(FICH$RES$NUM,.NT,1024,.CODEREP);
58    2           A$NT = .NT; NBR$RECU = 0;
60    2         END ECRITURE$DISQUE;
```

```
                $EJECT
61    1         CALL OPEN(.TINUM,.(':TI: '),1,0,.CODEREP);
62    1         CALL OPEN(.TONUM,.(':TO: '),2,0,.CODEREP);
63    1         CALL READ(1,.NT,255,.CTR,.CODEREP);
64    1         CALL OPEN(.FICH$RES$NUM,.NT,2,0,.CODEREP);
65    1         CALL FOOGO;
66    1         CALL NBMES(NB$MES$PTR);
67    1         CALL V9600;
68    1         A$NT = .NT;
69    1         NBR$RECU = 0;
70    1         NB$MESURES = 0;
71    1         COMPTEUR = OFFH;
72    1         CALL WRITE(TONUM,.(' '),1,.CODEREP);
                      /* DEMARRAGE ACQUISITION */
73    1         CAR = SPNPR;
74    1         DO WHILE CAR <> SPPR;
75    2           CAR = TTYIN AND 07FH;
76    2           IF PE
                  THEN DO; IF CAR = SPNPR
                           THEN DO; CAR = SPPR;
81    4                               PE = FALSE;
82    4                      END;
83    3                    ELSE DO; CAR = SPNPR; END;
86    3                END;
87    2           ELSE DO; IF CAR <> SPPR
                           THEN DO; CAR = SPNPR; END;
92    3                END;
93    2           CALL WRITE(TONUM,.CAR,1,.CODEREP);
94    2           IF CAR = SPNPR
                  THEN DO;CALL WRITE(0,.('SPECTRO NON
                          DEMARRE',RC,LF),21,.CODEREP);
97    3                END;
98    2         END;
99    1         CALL WRITE(0,.('SPECTRO DEMARRE',
                RC,LF),17,.CODEREP);
```

```
                $EJECT
100     1       LIAISON = FALSE;
101     1       DO WHILE NOT LIAISON;
102     2         CAR = TTYIN AND 07FH;
103     2         IF CAR <> ACK
                  THEN
104     2          DO;
105     3           IF (CAR = SPPR) OR (CAR = SPNPR)
                    THEN
106     3            DO; CALL WRITE(TONUM,.CAR,1,.CODEREP);
108     4            END;
109     3          END;
110     2         ELSE DO; LIAISON = TRUE; END;
113     2       END;
114     1       DO WHILE TRUE;
115     2         CAR = CAR AND 07FH;
116     2         CAR1 = CAR; ECRITURE = TRUE;
118     2         IF (CAR AND 07FH) = NAK
                  THEN DO; A$NT=A$NT-2;
121     3                  NBR$RECU=NBR$RECU-1;
122     3                  NB$MESURES=NB$MESURES-1;
123     3               END;
124     2          CALL WRITE(TONUM,.(STX),1,.CODEREP);
125     2         CAR = TTYIN; /* NUMERO MESURE */
126     2         IF (CAR1 = NAK ) OR (CAR1<> ACK)
                  THEN DO; A$NT=A$NT-2;
129     3                  NBR$RECU=NBR$RECU-1;
130     3                  NB$MESURES=NB$MESURES-1;
131     3               END;
132     2         ELSE DO;
133     3          IF (CAR1 = ACK)
                   THEN
134     3           DO;IF NB$MESURES=NB$MESURES$VOULU
                       THEN DO; CALL FIN;
138     5           END;
```

```
                $EJECT
 139    4           IF NBR$RECU = 512
                    THEN DO; CALL ECRITURE$DISQUE;
 142    5                END; END;
 144    3           ELSE DO; ECRITURE = FALSE;
 146    4                END; END;
 148    2         CALL WRITE(TONUM,.CAR,1,.CODEREP);
 149    2         IF ECRITURE = TRUE
                  THEN DO; CAR = TTYIN; RECU = CAR;
 153    3              END;
 154    2         ELSE DO; CAR1 = TTYIN;
 156    3                  CAR = CAR1;
 157    3              END;
 158    2         CALL WRITE(TONUM,.CAR,1,.CODEREP);
 159    2         IF ECRITURE = TRUE
                  THEN DO; A$NT = A$NT + 1;
 162    3                  CAR = TTYIN; RECU = CAR;
 164    3                  NB$MESURES=NB$MESURES+1;
 165    3                  NBR$RECU=NBR$RECU+1;
 166    3                  A$NT = A$NT + 1;
 167    3              END;
 168    2            ELSE DO; CAR1 = TTYIN;
 170    3                     CAR = CAR1;
 171    3                 END;
 172    2         CALL WRITE(TONUM,.CAR,1,.CODEREP);
 173    2         CAR = TTYIN; /* FIN DE LA MESRE */
 174    2         CALL WRITE(TONUM,.CAR,1,.CODEREP);
 175    2         CAR = TTYIN;
 176    2       END;
 177    1       END FFT;


MODULE INFORMATION:

     CODE AREA SIZE     = 03CDH     973D
     VARIABLE AREA SIZE = 0031H      49D
     MAXIMUM STACK SIZE = 0008H       8D
     183 LINES READ
     0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

3ème PARTIE

NOTATIONS

PU = unit central (central processor unit)

AM = mémoire vive (Random access memory)

OM = mémoire morte (Read only memory)

SART = interface de communication série programmable
(Universal Synchronoud Asynchronous Receiver/Transmitter)

PI = interface de communication parallèle programmable
(Programmable Peripheral Interface)

= système réalisé

BC = Carte processeur complet (Single Board Computer)

DS = système de développement (Microcomputer Development System)

= interféromètre de Michelson

PB = la carte processeur maître (Integrated Processor Board)

OC = la carte processeur esclave (Input Output Controler)

E = erreur de parité (Parity Error)

EMP = Emetteur vide (Transmitter empty)

RDY = Receveur prêt (Receiver Ready)

RDY = Emetteru prêt (Transmitter Ready)

- F F T A ␣ N° Fichier 1, N° Fichier 2 (CR)

  Exécution

  Spectro non démarré

  Après le départ

  Spectro démarré

  4096 points ⟶ sur Fichier 1

- PLM F T R ␣ Fichier 1 (CR)

  Attente indication (CR)

  FS = Fichier 2

  Verify ? (Y, N)

- CODAGE (CR)

  FE = ␣␣␣␣␣ (CR)

  FS = ␣␣␣␣␣ (CR)

  GAMME ?

- FFT 4 KD (CR)

  FE = ␣␣␣␣␣ (CR)

  FS = ␣␣␣␣␣ (CR)

- INTERP (CR)

  FE = ␣␣␣␣␣ (CR)

  FS = ␣␣␣␣␣ (CR)

  FENETRE = 1 à 5

  NUBL = 1 à 4 (CR)

  { interpolation, il écrit le bloc interpolé

  NUBL = Suivant ? (CR)

  pour sortir il suffit de taper 6

- EDITE 2 (CR)

  FE = ␣␣␣␣␣ (CR)

  NUBL = 1 à 4 existant (CR)

: LP : ? N (CR) (N, Y, D, E)

  DB = ␣␣␣␣␣ (CR)

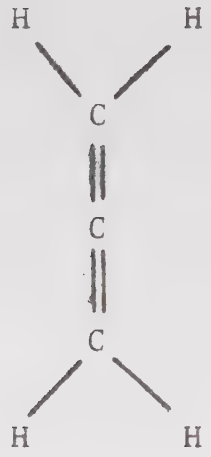
FN = (CR)

: LP : ? E fin bloc de 5120

NUBL = si 6 alors la fin

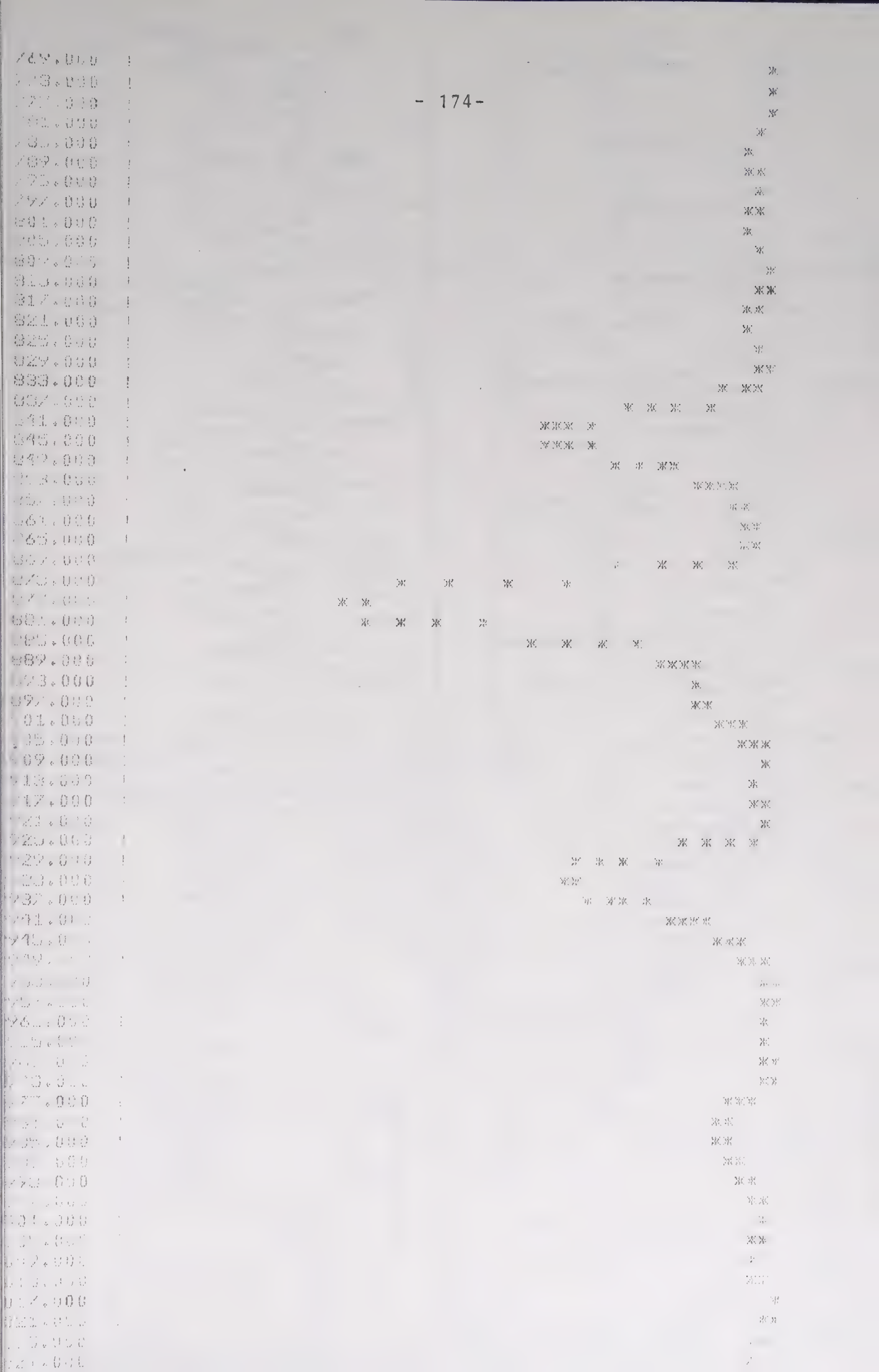
Le traitement à haute résolution avec un nombre de mesures égal à 4 K nous a permis d'avoir la courbe de la Figure C-4. Le tracé benson de la région équivalente à très haute résolution dont le nombre de mesures, enregistré sur la bande magnétique et traité CIRCE, dépasse 128 K.

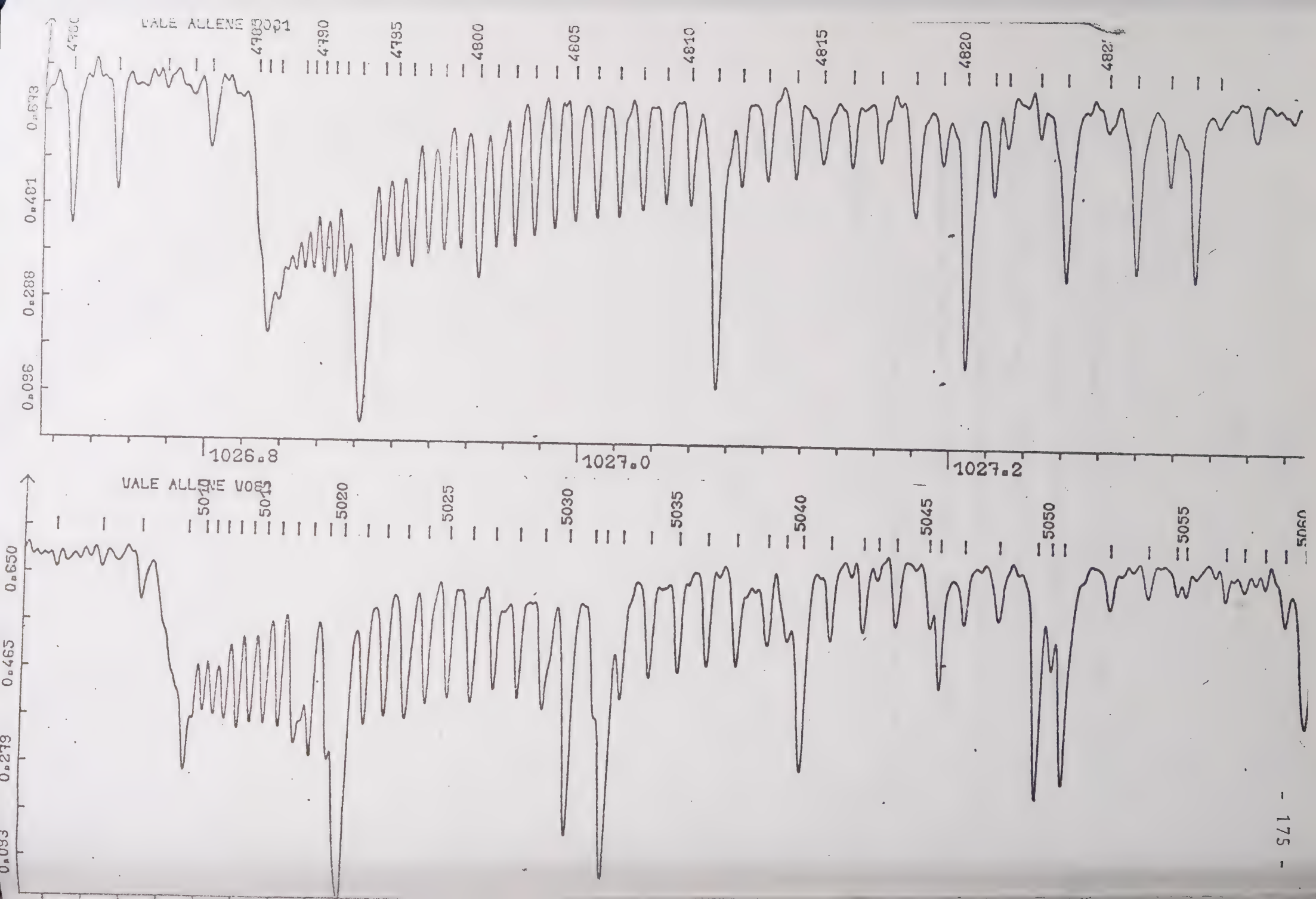
Ce spectre représente le spectre de la molécule d'ALLENE

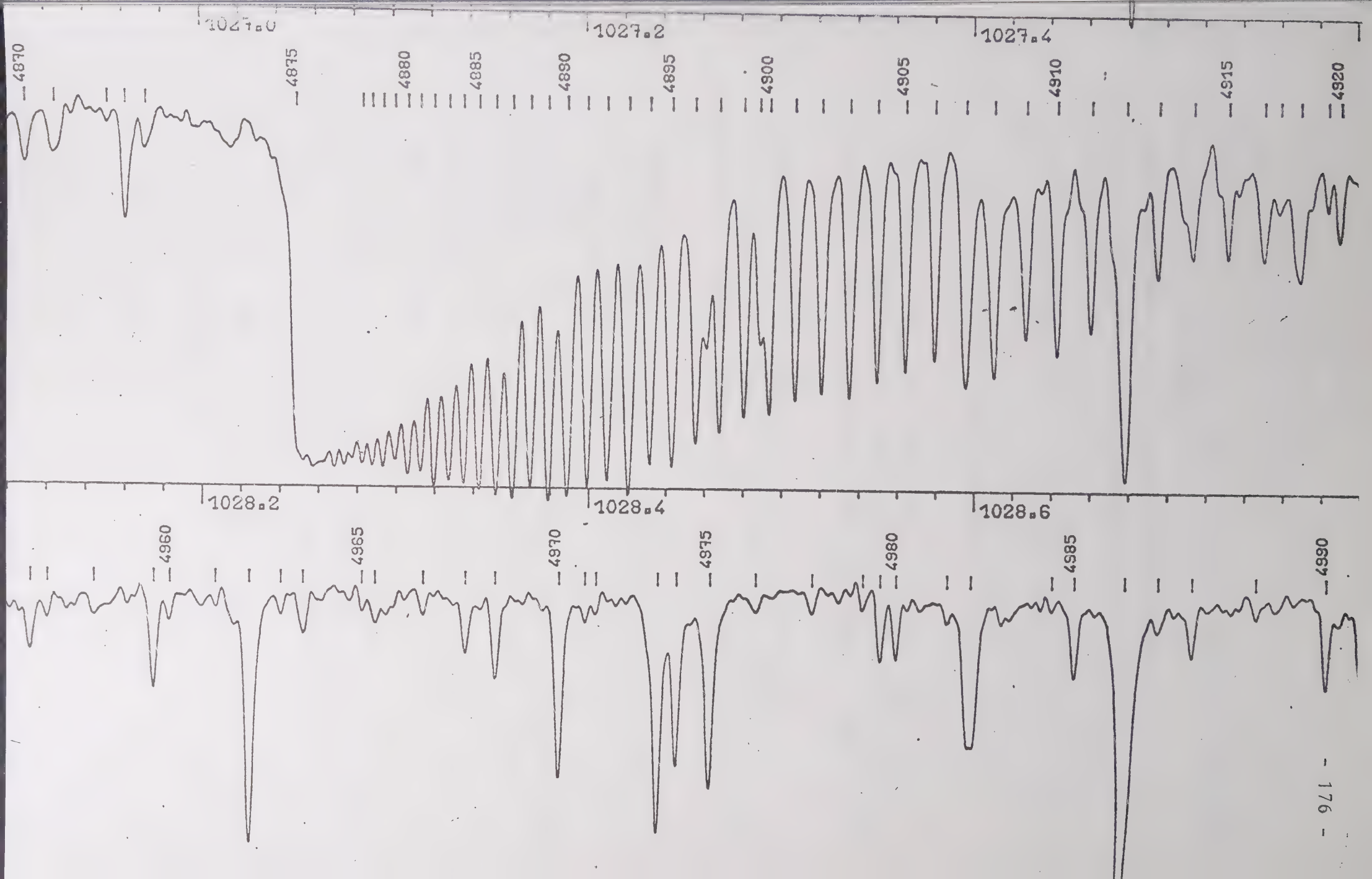


C = carbone

H = hydrogène




VALE ALLENE 0091
4785
4790
4795
4800
4805
4810
4815
4820
0.673
0.481
0.288
0.096
1026.8
1027.0
1027.2
VALE ALLENE V08
5020
5025
5030
5035
5040
5045
5050
5055
0.650
0.465
0.279


1027.0
1027.2
1027.4
4870
4875
4880
4885
4890
4895
4900
4905
4910
4915
4920
1028.2
1028.4
1028.6
4960
4965
4970
4975
4980
4985
4990

REFERENCES

[1] A. AHO et all. - The design and analysis of computer algorithms. Addison-Wesley, California, 1974.

[2] A. ANGOT - Compléments de mathématiques pour ingénieurs de l'électrotechnique et des télécommunications. Ed. de la Revue d'Optique, Paris, 1961.

[3] R. BOITE et H. LEICH - Les filtres numériques - Masson, Paris 1980.

[4] O. BRIGHAM - The fast Fourier Transform - Prentice-Hall, New Jersey, 1974.

[5] C. CARAISCOS, B. LIU, Two dimensional DFT using rixed Time and Frequency Decimations. Department of electrical Engineering and Computer Science, Princeton University, IEEE, ICASS, Vol. 1 1982

[6] V.V. CIZEK, Recursive calculation of Fourier Transform of discrete signal. Institute of Radio engineering and electronics Czechoslovak Academy of Sciences, Prague, IEEE, ICASSP, Vol. 1 1982

[7] W. CLIFFORD - Roundoff noise in floating Point fast Fourier Transform computation. IEEE Trans Audio and Electro, Vol. AU-17, N° 3, Septembre 1969.

[8] H.W. JOHNSON, C.S. BURRUS, The Design of optimal DFT Algorithms using dynamic programming. Rice University, Houston. IEEE, ICASSP, Vol. 1, 1982

[9] J. CONNE - Thèse, Université P. et M. Curie, PARIS VI, 1969.

[10] P. CONNES - G. MICHEL - "Interféromètre de voyage" - APPL. OPT.

[11] W. COOLEY et ALL. - The fast Fourier Transform algorithm : programming considerations in the calculation of sine, cosine, and Laplace transform - Journal of sound vibration, Vol. 12, pp. 315-337, Juillet 1970.

[12] W. COOLEY et W. TUKEY - An algorithm for the machine calculation of complex Fourier series - Math computation, Vol. 19, pp. 297-301, Avril 1965.

[13] G.D. COVERT, B.L. TROUTMAN. "A 32 point monolithic FFT Processor chip", TRWLSI Product, IEEE, ICASSP, Vol. 2, 1982.

[14] H. DELOUIS - Thèse, Fac. des Sciences d'Orsay, 1973.

[15] A. GHERBI - Thèse, Université P. etM. CURIE, Paris VI, 1979.

[16] J. GOOD - The interaction algorithm and pratical Fourier series. J.L. Roy. Stat. Soc. Ser. B, Vol. 22, pp. 372-375, 1960.

[17] B. NEW, D. BRAIN, "Bipolar VLSI Facilitates Fourier Transformation", Advanced Micro Devices - M/S 5, IEEE, ICASSP, Vol. 2, 1982.

[18] MAILLARD, Thèse - Orsay 1967.

[19] D. MARKEL - FFT Pruning, IEEE trans. and elect., Vol. AU-19, N° 4, Décembre 1971.

[20] J. PINARD, Thèse 3ème cycle, PARIS 1963.

[21] E. ROUBINE - Introduction à la théorie de communication. Tome 1, Masson, PARIS 1970.

[22] G. ROUCAIROL - Thèse, Université P. et M. CURIE, PARIS VI 1978.

[23] C. SHANNON et W. WEAVER - The mathematical theory of communication - Univ. Illinois Press, p. 53, 1949

[24] C. SINGLETON - Algorithm 338 Algol Procedures for the fast Fourier transform - ACM, Vol. 11, pp. 773-776, Novembre 1968

[25] K. TOYOHISA et B. LIU - Accumulation of round off error in fast Frourier Transform. Journal of the assoc. for computing machinery, Vol. 17, N° 4, pp. 637-654, Octobre 1970.